Software Design 中文版01

（日）技术评论社 编

机器学习+数据分析

Software Design 01

中文版

人民邮电出版社
POSTS & TELECOM PRESS

数字版权声明

图灵社区的电子书没有采用专有客户端，您可以在任意设备上，用自己喜欢的浏览器和PDF阅读器进行阅读。

但您购买的电子书仅供您个人使用，未经授权，不得进行传播。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

图书在版编目（ＣＩＰ）数据

Software Design中文版01 / 日本技术评论社编 ；
苏祎译. -- 北京 : 人民邮电出版社, 2014.3
ISBN 978-7-115-34705-3

Ⅰ. ①S… Ⅱ. ①日… ②苏… Ⅲ. ①软件开发 Ⅳ.
①TP311.52

中国版本图书馆CIP数据核字(2014)第034159号

内 容 提 要

Software Design 是日本主流的计算机技术杂志，旨在帮助程序员更实时、深入地了解前沿技术，扩大视野，提升技能。内容侧重于网络、操作系统、开源软件和信息处理技术等。

本期的主题为：机器学习与数据分析。第1特辑详细讲解了日趋重要的数据分析技术，以及 Excel、R、Mathout 等数据分析工具的使用方法。第2特辑分别介绍了面向PC和面向服务器的基准测试技巧。这对开发者评估硬件性能、发现系统瓶颈大有裨益。本书适合各行业软件开发者阅读。

◆ 编　　　[日] 技术评论社
责任编辑　乐　馨
执行编辑　武晓宇
责任印制　焦志炜
装帧设计　broussaille 私制
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　　　　　　　印刷
◆ 开本：787 × 1092　1/16
印张：10
字数：246 千字　　　　2014年3月第1版
印数：1 - 4000 册　　　　2014年3月北京第1次印刷
著作权合同登记号　图字：01-2014-0495 号

定价：39.00 元

读者服务热线：(010)51095186 转 600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

目录

Software Design 中文版 01

Contents

分栏目录

Software Design 中文版 01

Contents By Topics

特辑1 Special Feature 01

让我们重新审视数据的价值

学习数据分析，从这里开始

特辑2 Special Feature 02

是否拥有自己的评价标准？能否发现系统性能瓶颈？

基准测试应用技术

专　栏

软件开发

操作系统/网络

专家视点

2

第55回

IT工程师必须知道的

最新术语

文/杉山贵章 Sugiyama Takaaki（ONGS公司）
takaaki@ongs.co.jp
译/苏祎

征集主题

本连载正在征集最近大家关注的名词和今后准备研究的主题。如有兴趣，请发邮件至sd@gihyo.co.jp。

Open Cloud（Ⅱ）

上一期中，我们介绍了逐渐在业界普及开来的OpenCloud的动向及主要的平台软件。本期作为上期的续篇，介绍一下在PaaS层和网络层的平台软件。

Cloud Foundry

Cloud Foundry是以VMware公司为开发主力的开源的PaaS平台软件。Cloud Foundry不仅能作为独立的软件来使用，而且VMware公司还提供了使用Cloud Foundry搭建而成的公共云服务CloudFoundry.com，并且还把VMware虚拟机和Cloud Foundry环境打包成一个名为Micro Cloud Foundry的软件。

Cloud Foundry的架构特征是可根据功能明确区分各个组件。应用的组成管理、Web服务API等核心功能运行在被称为Cloud Controller的模块上。DEA（Droplet Execution Agent）模块提供了应用的运行环境。外部请求经过Router或负载均衡分发给合适的DEA。在应用或DEA发生未知错误时，它具有可以自动重启实例的恢复功能。

Cloud Foundry的使用许可是Apache License 2.0，支持的编程语言为Ruby、Java、Node.js、Scala、Erlang等，支持的数据存储为PostgreSQL、MySQL、MongoDB、Neo4j、Redis等。

OpenShift

OpenShift是Red Hat公司PaaS平台软件及服务的总称。OpenShift的提供形式包含：开源软件OpenShift Origin；Red Hat公司提供的PaaS服务OpenShift Onine；面向企业的PaaS服务OpenShift Enterprise，这项服务提供商业支持的预置OpenShift环境。

OpenShift的架构大致由两种模块构成：提供应用部署和各种数据的存储、认证功能的Broker；托管应用及服务的Node。Node以Gear为单位，管理应用的运行环境。Gear的最大优势在于其可以自由地新建或删除，因此可以根据需要灵活扩展环境。

OpenShift Origin以Apache License 2.0的形式提供，支持的编程语言为Java、Ruby、Node.js、Python、PHP、Perl等，支持的数据存储为PostgreSQL、MySQL、MongoDB等。

SDN/OpenFlow

SDN（Software Defined Networking，软件定义网络）是使用软件管理网络组成技术的总称，它逐渐成为在构筑云平台时不可或缺的东西。其中，开放网络基金会（ONF）推进标准化的OpenFlow，是备受瞩目的技术。

OpenFlow的网络就是由多个OpenFlow交换机和集中管理这些交换机的OpenFlow控制器组成的。OpenFlow控制器定义了如何计算包的路径以及收到包后的处理方式。OpenFlow交换机基于上面的定义，进行包的传送。这样就不用逐个对交换机的设置进行变更，只要设置好控制器，就能管理整个网络结构了。

除了OpenFlow外，SDN还有Linux基金会在推进的OpenDaylight项目。这个项目意在在SDN里构建新的OSS平台，它将不仅包括网络控制器，其构想还将囊括编程界面和网络协议第4至第7层的功能，以及网络虚拟化等技术。



网络工程师修炼手册

自己家的服务器机架之推荐篇

文/tomocha（http://tomocha.net/diary/）
译/阿逸
插画/高野凉香

第3回

机架的电源问题（1）



引　子

这次的主题是电源，我们将对断路器和签约电力等进行说明。①

什么是签约电力

当用电量超过了与电力公司签约的容量，断路器会落下，即跳闸。这个断路器称为电流断路器，是电力公司和用电客户根据用电合约设置的装置，当回路中的电流超过合约规定的安培时，断路器就会切断供电回路。一般来说，最大供电量在大多数情况下是固定的。最大供电量越多，基本费用就越高。若更改供电合同规定的供电量，则需要更换电流断路器。根据断路器的颜色或标注的说明，或者查看电力公司提供的账单或发票，都可以了解最大供电量是多少（见图1的左侧，最大供电量为60A）。

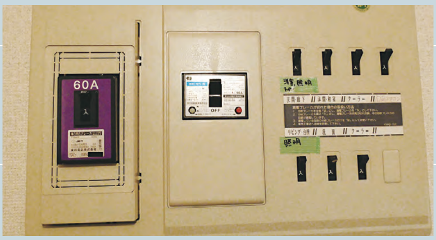
一般我们需要根据最大供电量，选择合适的设备。在搞清楚日常生活需要多少用电量的前提下，再考虑最大供电量应该为多少。同时使用的电力必须在最大供电量的范围之内。例如，独居日常生活需要30A（安培）左右；和家人一起居住的独栋别墅或者两居室公寓的话，一般在40A到60A之间。空调、电吹风、微波炉等同时使用时如果出现跳闸，那就说明最大供电量还不够（安全断路器跳闸的情况除外）。在这种情况下，就需要重新考虑用电方法，并提高和电力公司签约的最大供电量了。

用户可以向电力公司询问它能够提供的最大供电量是多少。一居室的情况下，由于供电设备的限制，电力公司一般只能提供30A到40A。并且需要注意的是，电力公司一般很难向公寓住宅提供7kVA（相当于70A）以上的供电，即使可以也需要额外进行电力设备施工。

什么是安全断路器

可能大家有过这样的经历：一个插座上连接的电器过多，或者房间的墙插上连接的电器过多时，就容易跳闸。例如，一边看电视一边使用电吹风和电热水壶的时候，只听到“啪”的一声墙插的供电就中断了，但房间的照明电路仍正常工作。其实，这是安全断路器跳闸了。一般来说安全断路器的安全电流为15A或

图1　60A的断路器，旁边的小开关是安全断路器



① 文中解说的是日本的情况，与中国的实际情况有所不同，请读者注意。——编者注

者20A，这个数值依赖于从断路器到插座之间的电线和插座的具体情况。

由于电线也存在电阻，电流流过的时候会产生热量。过大的电流会造成电线发热，有可能引发火灾，所以需要设置安全断路器（见图1右侧，每一个回路对应一个20A的安全断路器）。

根据一般的室内设计，厨房中经常会摆放冰箱、微波炉、电饭煲等大用电量的电器，所以多数情况下会为厨房配备一路独立回路。同样，卫生间的电路也是独立的，因为考虑到会在其中使用烘干机、洗衣机等。另外，空调的供电插座一般是独立的15A～20A的回路。

没有事先设想到会在房间的墙插上使用大功率的电器，或者是受限于回路的数量，将多个房间的墙插串联在一个回路中，这样的情况也是存在的，也就是说，没有以房间为单位配备独立的供电回路的情况并不少见。要搞清楚供电回路的连接，可以将照明器具连接在插座上，通过开关断路器来确认。公寓的供电回路一般很难改动，所以选择住宅的时候就要选择供电回路符合要求的房屋。有时，如果客厅很大的话，可以分两路回路来供电。

应该如何处理服务器机架的插座?

比较好的做法是从安全断路器单独引一回路到机架供电，这么做的理由是为了尽可能地减少外部电路对服务器的影响，避免因错误地连接电器导致的跳闸或电压下降，以及电器故障对电路产生的影响等。

如果每间房间都配有安全断路器的话，那就更方便了，每个房间使用一条回路就相当于15A的专用供电。并且，根据房屋和建筑物的不同，存在为空调专门铺设供电线路的情况。这种线路能够承载15A～20A的电流，但需要注意，其电压可能是200V的。

接着，让我们来计算一下机架上放置的各类设备究竟需要消耗多少电力。24口或者48口的L2交换机一般在100W以下，家用上网设备一般在10W以下，还要考虑为机架配备的高性能L3路由器以及别的用电设备的用电量。网络设备的耗电量基本符合规格说明，虽然最近为了节约能源出现了将不使用端口的电源关闭的eco模式，但它不会像服务器一样出现用电量的巨大波动，所以用电量还是比较容易计算的。

近期，服务器的电源管理有了长足的改善，CPU和I/O根据负载情况，用电量会有很大的变化：空闲状态下测出的功率不到100W，高负载的瞬间功率消耗接近200W，这样的情况并不少见。配置有多台服务器的情况下，高负载时断路器跳闸也是很有可能的，所以供电量要有一定的冗余。

运气好的话，不需要额外的施工就能够确保一间房间30A左右的供电。然后就是要做好电器的接地工作。一般家用的插座都是两孔，很少有接地线。厕所、厨房的插座肯定有接地线，空调的插座一般也有接地线。所以我们需要将电器正确连接插座，确保电器接地。

总　结

一般来说日常生活所需的电力在20A到30A之间，网络设备和服务器需要长时间工作并消耗电力，因此两者相加就是需要向电力公司申请的耗电量。否则，通常一两个半幅机架的设备运转，大多数的情况下会出现供电不足了。如果继续配置更多的机架的话，那就变成只能放置设备的空架子了。

最后，需要注意的是，断路器和电路的铺设施工需持有“二级电气工程师”的资格。

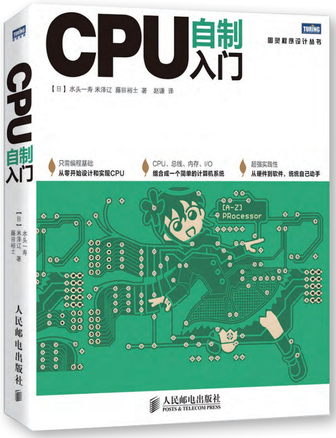
玩转自制，自我进化

- 从零开始设计和实现
- 兼具趣味性、实用性与学习性
- 自己动手揭开计算机技术的神秘面纱



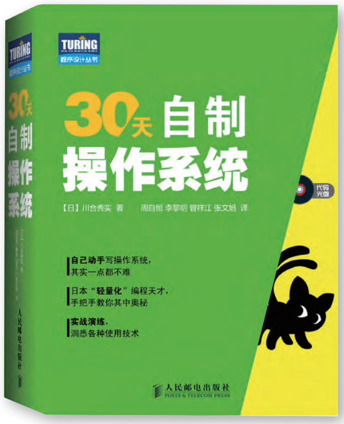
书名：自制编程语言
书号：978-7-115-33320-9

《自制编程语言》手把手地教读者用 C 语言制作两种编程语言：crowbar 与 Diksam。crowbar 是运行分析树的无类型语言，Diksam 是运行字节码的静态类型语言。这两种语言都具备四则运算、变量、条件分支、循环、函数定义、垃圾回收等功能，最终版则可以支持面向对象、异常处理等高级机制。所有源代码都提供下载，读者可以一边对照书中的说明一边调试源代码。这个过程对理解程序的运行机制十分有帮助。



书名：CPU 自制入门
书号：978-7-115-33818-1

《CPU 自制入门》利用 FPGA，为读者开启了一个崭新的自制 CPU 的世界。全书分为 3 章，分别介绍计算机系统最底层的 3 个部分：CPU 设计制作、电路板设计制造以及汇编编程。将如此广泛的技术内容以实践的方式融合成一册，该书可谓首屈一指。本书可以帮助软件工程师深入了解硬件与底层，开发出高效代码。硬件工程师可以在本书基础上设计定制硬件，开发高速计算机系统。



书名：30 天自制操作系统
书号：978-7-115-28796-0

《30 天自制操作系统》是一本兼具趣味性、实用性与学习性的书籍。作者从计算机的构造、汇编语言、C 语言开始解说，让读者在实践中掌握算法。在这本书的指导下，从零编写所有代码，30 天后就可以制作出一个具有窗口系统的 32 位多任务操作系统。以课题为主导，边做边玩，抛开晦涩难懂的语言，行文风格十分随性，还充满了各种欢乐的吐槽，适合操作系统爱好者和程序设计人员阅读。

新书预告



书名：自制编译器

本书用 Java 实现了一个 C_b（一种 C 语言的精简版）的编译器，可以将 C_b 程序编译为 Linux 上原生的 ELF 格式可执行文件。



书名：两周自制脚本语言

这是一本绝佳的编译原理入门书。本书从编程语言的原理讲起，从零开始教读者在 Java 虚拟机上按照编译过程自制脚本语言 Stone，并深入讲解编程语言的编译机制。

新潮数码 DIGITAL GADGET

文 / 安藤幸央　Yukio Ando（EXA公司）

Twitter：@yukio_andoh
个人主页：http://www.andoh.org/

译 / 苏祎

Volume 175 》》》

Google I/O 2013开发者大会上出现的数码设备

主题演讲中感受到的未来蓝图

美国时间2013年5月15日至17日的3天，Google I/O 2013开发者大会在旧金山召开。Google I/O是面向开发者的大会，针对Google的各项技术设有多个分会场。大会今年吸引了6000人，是会场所能容纳的最多人数，世界各地通过网络收看或收听的参加者超过了100万人。今年的分会场数量达到189个，展台170个，均超过了去年的水平。如此规模的大会，不仅吸引了硅谷周边的开发者们，更吸引了很多来自欧洲、亚洲等地的参加者。

如果用一个词来概括对今年大会的印象，“踏实”比“华丽”更为合适。在首日的主题演讲中，可使现在生活更加便利的踏实而又明晰的新技术以及技术的改良得到突出，而那些华丽的未来技术退居其次。

大多开发者原本期待的Android话题在整体中占的比例较低（之后再详述），而去年在Google I/O上成为焦点的眼镜式穿戴设备兼相机的Google Glass，又在今年以另一种形式成为了热点。

会场中，除了Google公司的员工外，已有许多其他人佩戴了Google Glass。他们是在去年的Google I/O上预约购买，并迅速拿到手的用户。但是在主题演讲中，并没有提到任何与Google Glass相关的话题，似乎让人感受到Google的声音检索及Google Now的进化是推广Google Glass前置的两步棋。给人的整体感觉是，Google Glass还未做好进军大众市场的准备，希望开发者能来扩充其服务内容。

在主题演讲的最后，因声带疾病而在疗养中的Google CEO拉里·佩奇登场，让一直担心的大家放下心来。他用稍显沙哑的声音，缓慢地向现场观众说道：“希望不要总拿Google与其他公司比较，我真地不觉得有趣。不能因为消极的态度阻碍了人们获得新的成果。我们应当创新，并非每一项新技术都是零和博弈。”

这次主题演讲中发布的内容，与其说是创新，不如说是对既存产品的延伸。从侧面上也可以看出Google作为一家公司将人力和资金都投向了能确实获取利益的方面，特别集中在以Google Maps为开端的GEO（地图及位置信息关联）的改进及推广。

在会场进行的传感器实验

此次Google I/O大会初次尝试



此次会场的Moscone中心。Maps的Pin的设计也更新了。



会场入口处巨大的Google I/O标志。



Google街景的摄影器材。（左：适用于山地；右：适用于海底）

了在会场中铺设传感器网络。这是由Google Cloud团队与O' Reilly Data Sensing Lab共同为此次大会制作的系统。基于开源硬件Arduino的525台环境传感器遍布会场各个角落。为保障各个环境传感器之间的数据通信，构建了以XBee为基础的Mesh网络。该实验是把传感器中暴露在外的主板，与Google Cloud Platform的软件连接，将会场中的温度、湿度、声音分贝、楼层的震动、空气质量、灯光亮度等信息作为数据收集并实现可视化。

比如说，若希望在会场中小憩一下，或打一个电话时，就可以使用它来查找安静的场所。这个实验使得Google"整合全球信息，使人人皆可访问并从中受益"的使命变得更为现实具体。

围绕Android的状况

主题演讲中，Google发布了Android软件的新开发环境Android Studio，它是以大家熟知的Java开发环境IntelliJ为基础搭建的。虽然Android的生态环境变得越来越巨大，但胜者与败者却已被明确划分开来。尽管如此，现在的胜者并非一劳永逸，不知不觉被后来者赶上也不是一件新鲜的事。并且，Android设备的硬件、OS、软件、框架、用户界面等要素的复杂结合，使得单纯通过软件去创造一些新的内容变得更加复杂。

另外，虽然此次大会并没有明确表达，但是可以感受到的是，尽管现在Android的身影随处可见，但今后（原生的）Andorid应用将逐步消失。这也可以从Google内部的人事变动看出来。Android部门的负责人从原来的Andy Rubin变成了负责Chrome的Sundar Pichai。

虽然不会是近期，但未来Android很可能被集合了Web应用与浏览器的Chrome OS统治。此次的Google I/O大会中，对于Chrome Packaged Apps及Web组件的呼声也很大，浏览器未来将可能作为操作环境，成为类似Chrome Runtime的定位。Java诞生以来，大家都在追求一种梦想，试图让各种技术在任何环境中都可以运作，Web应用与Chrome似乎也开始追求这个目标。

巨大的会场与趣味横生的各种会议

189个分会场会议将Google的各种技术详细介绍给开发者、工程师和设计师。今年Chrome/Web关联的会议非常多。会议的内容不仅仅是讨论Web及Android智能手机用户界面的制作方法、思考逻辑等，还延伸到在这背后的认知心理学应用方法、车载应用的展开等更广更深的层次。

Google I/O大会之后各个会议的视频立即在YouTube上公开。（配合英文字幕可以更好地理解，也可以通过机器翻译查看日语字幕。）

Google I/O 会议视频一览：
https://developers.google.com/live/io/

1层至3层的巨大会场比想象的还要大，除了展台以外，还有可供放松休息的区域，以及可以直接向Google开发者提问的"工作时间"等供开发者们互相交流的场所。

Google的魔法

在Google I/O大会上，设有一些场所，可以向在Google工作的人（他们被称为Googler）询问现场的情况，也可以在"工作时间"直接提问，或提出自己的意见。

我们了解到，在Google这么多的有知识、有能力的人，其实就是在专业的环境下，游刃有余并且高效地做着极其普通的事情，并没有什么特别的魔法。但是Google人才众多，大家见识极广，令人敬佩不已。各路专家齐聚一堂，营造出可以不断创造新内容并解决问题的环境，而且可以一直维持着这种氛围。

更广泛的活跃空间

此次大会令人印象深刻的是，除了现在已有的众多服务与先进



位于山景城的Google Store的临时店铺。



会场中的展台场景。图中显示的约是整体面积的八分之一。



空中漂浮着的Google飞船。可以查看到其搭载的相机拍摄的画面。



运用了大量Chromebook Pixel的显示器。

技术外，还有许多未能开拓出的点和必须解决的课题。日本的开发者及设计师可以挑战的课题和发挥的空间也还很多。但是，与世界科技发展的广度、深度以及信息的速度相比的话，我们还十分落后。

把自己定位为工程师还是设计师，这样的区分是没有意义的。工程师也需要学习设计相关的知识，而设计师也可能要写代码。最初可能两者都无法达到很高的水准，但通过不断的努力，将会有一天能越过以往无法翻越的高墙。工程师可能无法创造出一流的视觉设计，但会更加理解设计师的内心想法及坚持的信念。设计师可能无法写出一流的代码，但也能理解工程师的感受及逻辑。这样的相互理解与沟通的效果是巨大的，它能使有广泛技能的专业人才获得更广泛的活跃空间。

Google I/O大会等各种面向开发者的大会，都会立即把各种消息公开在网上。但还有许多事情，如智能手机和平板电脑在人们的生活中得到了很广泛的使用，特别在美国的城市地区，Google Maps、Foursquare、Facebook 都已经深深扎根于日常生活中，这些热烈的气氛及人与人之间的交流是需要到现场才能了解到的。今年的Google I/O大会的确非常充实。



会场的525个角落设置了主板外置的环境传感器。

gadget 1

NVIDIA SHIELD

http://shield.nvidia.com

游戏专用的Android设备

著名图形芯片厂商NVIDIA推出了最新的搭载了Tegra 4，Android 4.2.1的游戏专用机SHIELD。该设备配备了HDMI输出、5英寸、1280×720多点触控液晶屏，以350美金的价格从5月20日起接受预订，计划6月开始出货。SHIELD重579 g，与其外观感受一样略微粗犷。难以让人察觉其为Android OS的设备，游戏专用机的印象更为强烈。



gadget 2

Google Chromebook Pixel

http://www.google.com/intl/ja/chrome/devices/chromebook-pixel/

超高精细浏览器专用笔记本电脑

2560×1700像素的分辨率、400 nit[①]的亮度、178度广角的液晶屏的Google Chromebook Pixel。与最新型的轻薄笔记本电脑相比，1.52 kg略显沉重，但其优势是，拥有5小时电池续航时间，通过Google账号登录可以立刻使用。北美版的机型可以搭载LTE SIM及SSD，并附赠1 TB的Google Drive的使用权。键盘的配置可以变更。价格为1299美元。



gadget 3

Rethink Robotics BAXTER

http://www.rethinkrobotics.com/

模仿机器人

Rethink Robotics BAXTER是一通用的用于生产的机器人。通过教给它人类的动作，它可以反复地进行该动作。脸部的显示器会显示出可爱的眼睛和表情。该产品出自Rodney Brooks先生的公司，之前他就因发明清扫机器人Roomba而闻名。产品的亮点是一般人通过手把手地教授机器人动作，就可以完成似乎非常复杂的设置了。价格为22 000美元起。



gadget 4

Oculus VR

http://www.oculusvr.com/

游戏专用的虚拟现实头戴式显示器

Oculus VR是通过双目可追踪立体视觉，具有运动功能的头戴式显示器（HMD）。与以往的HMD的最大区别是左右90度的视野。分辨率为640×800像素的两块屏幕左右拼接，惊人的沉浸感和轻便感，超低延缓的6轴追踪传感器带来了优秀的追踪性，可以期待它的问世会带来新型游戏的登场。输入模式为DVI/HDMI及USB连接。价格为300美元。会场中Recon公司的HMD也大受欢迎。



① 尼特，亮度单位，1尼特=1坎德拉/平方米。

文 / 结城浩（Hiroshi Yuki）
http://www.hyuki.com/
twitter: @hyuki
译 / 成勇

结城浩的再发现随想



Threshold

Threshold——阈值

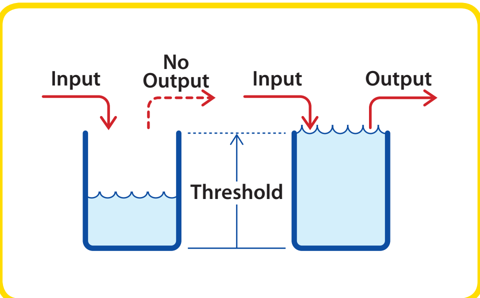
阈值是什么

用杯子从自来水管接水，水会积存在杯子里。在水溢出杯口之前，杯子外面不会湿。一旦溢出，杯子外面就会湿掉（图 1）。这次我们谈论的，就是和这个现象有关的话题。

阈值（Threshold）是指：在超出这个值以前，什么都不会发生，超过了这个值，就会发生变化和反应的一个值。以图 1 的杯子为例，不会弄湿杯子外面、能在杯子中保存的水量就是阈值。

带有阈值事物的输入（Input）和输出（Ouput）的关系，可以用图 2 来表示。在输入超出阈值以前，不会有任何输出。

▼ 图 1　用杯子表示的阈值示意图



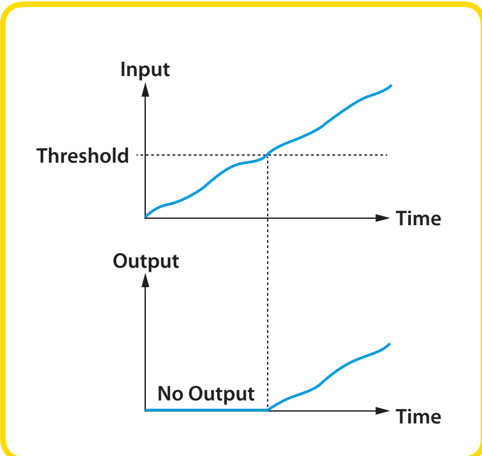
数字化和阈值

计算机是一种使用开（ON）和关（OFF）两种状态进行计算的数字设备，这个开和关，在计算机内部是使用电压来表示的。为了将模拟信号的电压，转换处理成数字的开和关，就要用到阈值。也就是说，如果超出某个值，就当成开（ON）处理；如果没有达到这个值，就当成关（OFF）处理，这个值就是一个阈值。

两个阈值

在达到阈值以前，输入的变化对于输出没有任何影响。但是，当输入达到阈值附近时，输入的波动将会影响输出。

▼ 图 2　带有阈值事物的输入和输出的关系



比方说，识别开关的阈值是0.5 V，如果电压一直在0.5 V附近上下波动，与此相对的数字值也会在开和关之间波动，如图3所示。

为了防止输出不稳定，我们可以设置“关变成开”和“开变成关”两个阈值。这样，阈值就可以在开和关之间保持稳定，而不会波动，如图4所示。这样即使输入不稳定，输出的波动也会减小了。

情感表达的阈值

个人情感的外露似乎也存在阈值，这就是情感表达的阈值。对一个人而言，压力、工作量可认为是输入，情感的外露可以认为是输出。

忍耐力强的人（情感表达阈值高的人），即使压力很大也不会把自己的情感挂在脸上，但是忍耐力低的人（情感表达阈值低的人），稍有压力，就马上能从脸上看到。

当然，情感表达阈值高些好还是低些好，不能一概而论。忍耐力强的人，干起活儿来雷厉风行，但是别人很难了解这个人真正的状态如何。

拐点阈值

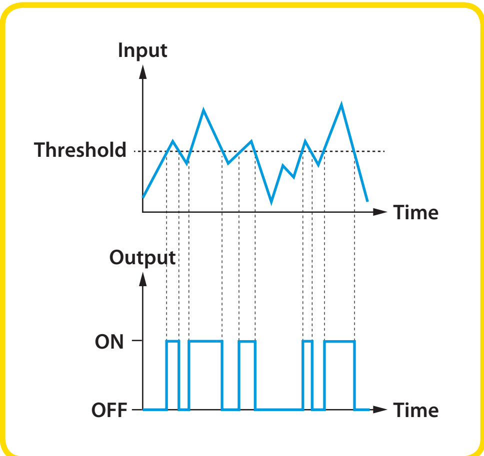
在商品销售上，有种现象称为拐点。刚开始在宣传上不断投入成本（输入），但是销售（输出）一直上不去。但是，在某个时间点销售会突然激增。这个拐点可以说就是阈值。

因为有阈值的存在，所以输出不会随着输入的变化而变化。如果想尽早知道，察觉变化的话，需要降低阈值，或者观察中间性的输出。例如，在销售实现输出前，看看Web页面的页面点击数，调查闲谈中谈及商品的次数等中间性的输出。

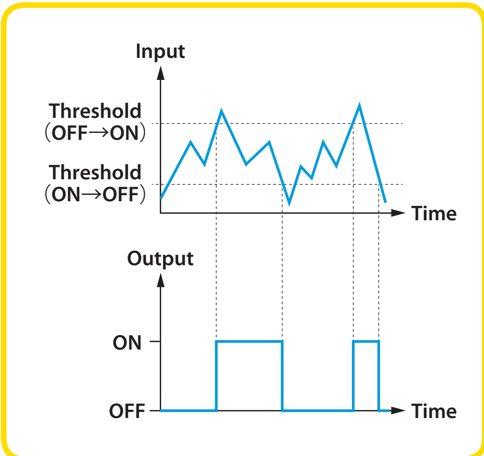
◆　◆　◆

你的身边有没有可以称为阈值的东西？它的输入和输出都是什么？请试着思考一下吧。

▼ 图3　输入波动对输出的影响



▼ 图4　用两个阈值防止输出波动



结城浩（Hiroshi Yuki）

生于1963年，日本资深技术作家和程序员。在编程语言、设计模式、数学、加密技术等领域，编写了很多深受欢迎的入门书。代表作有《数学女孩》系列、《程序员的数学》等。

作者网站：http://www.hyuki.com/

enchant
～激发创造力的魔法～



文/清水亮（SHIMIZU Ryo　Ubiquitous Entertainment Inc股份有限公司）　译/印勇刚

URL http://www.uei.co.jp

第3回　在秋叶原创建NASA

当田中谅开始制作日后被称做enchant.js的时候，我正和近藤诚、镰田淳二、辻秀美一起前往旧金山，伏见辽平已于先期到达。我们的目的是参加被称为GDC（Game Developer Conference）的游戏开发者大会。

太平洋上空的编程冲刺赛

这一天清晨，我们在京成线的上野站会合，乘坐开往机场的光线号（skyliner）电车。刚刚上车不久，近藤就对我半开玩笑似地说："清水先生，这次我们在飞行途中，该做些什么呢？"

说来话长，那时近藤还是学生，他在我们这里兼职。在前往旧金山参加MacWorld的飞机上，他中了我的激将法，当场奋起，编写应用程序。这个应用的目标是让用户在iphone、ipad上，能够体验使用毛笔书写的乐趣。结果，就在飞行的过程中，这个应用就干净利落地完成了，而且让人出乎意料地满意。

飞机一抵达旧金山，我们就再接再厉、一鼓作气，使用"i书道"的名字正式发布了这款应用。随后的反响也很令人满意，"i书道"不仅在日本的新年期间，跃居App Store排行第1名。后来，还被Apple的广告采用了，其LOGO"書"在世界范围内的Apple Store中受人瞩目。说到这个"書"的LOGO，那是我在飞机中，尝试着使用还处于编写过程中的"i书道"书写出来的呢。

也许正是因为有这样一个故事，所以近藤才开了一个"这次再做些什么好玩的"的玩笑吧？我当时正在关注HTML5的种种可能性，就干脆顺水推舟，提了一个有些疯狂的奇特方案：

"这次的出差人员都是UEI（Ubiquitous Entertainment Inc）（本文作者清水亮的公司）中最顶级的程序员，我们来比一比，看谁能在8个小时的飞行途中开发出10个游戏！"

"这……清水先生，您也参加吗？"

我平时一直喜欢只动嘴说大话，但这次也许不应临阵逃脱。虽然有"败下阵来、颜面尽失"的风险，但我的自尊心还是使我奋起迎接挑战。"那还用说嘛，我当然参加啦。现在就开始比赛，结束时间就是飞机着陆那一刻！"

我一声令下，所有人都开始编写程序了。为了保全作为上司的荣誉，我全力以赴。然而，当我编写到第6个游戏时，苹果电脑的电池耗尽了。真是很不甘心啊。结果，这场比赛，谁也没能写完10个游戏。

我们意犹未尽，后来进行了第2次比赛，目标是"12个小时内完成10个游戏的开发"。这样的比赛就叫做"程序员的短跑冲刺赛！"（其中，这场在飞行途中的突发性比赛，被我们冠名为"环太平洋上空的迷你游戏开发奥林匹克"，比赛的视频也都发布到了互联网上[①]。）

① http://www.youtube.com/watch?v=Z6B5bgWtPew

没有比编程更有趣的了

此次比赛（第2次比赛），我成功完成了13个游戏的开发。不过，之后我没有再参加，而伏见辽平在旧金山和我们会合后，却投入了比赛。对于本职工作为策划的辻秀美，我曾不假思索、信口开河地说：“我来教你，你也来编写游戏吧。”结果第2天，辻秀美便完成了一款相当不错的游戏！

随着“程序员短跑冲刺赛”的多次进行，各种新奇的构思不断涌现出来，着实令人惊喜。特别是镰田淳二那些奇奇怪怪、匪夷所思的想法，经常让众人爆笑不止。伏见则以非凡的技巧，开发了高水平的游戏。

实在是太开心了！编写程序的这种爽快劲儿，我好久没有体会到了。从构思到编程，与大家分享，获得大家的肯定，最后收获了欢笑和喜悦。

让程序员成为英雄

如同NICONICO视频一样，GDC大会是一个可以自由发表原创游戏的地方。只要是投稿的游戏，就可以让朋友们和自己一起来玩，也是一个自由交流感想的好地方。而使用HTML5，让这一切更具有可行性，特别是在智能手机上，尤为简单。

回国之后，我也要举办这样的活动。要让更多的人了解编程的快乐，如果能让程序员成为大家心目中的英雄，那将是一个多大的飞跃啊。

想到这里，我的脑海中浮现出了一个英文单词——leap，意思是“飞跃”。如果直接使用leap进行商标注册有点困难。于是，我准备在前面加一个数字9，改为9leap。这个9也是有含义的，当数字9加上1后，数字就会进一位。这意味着无论是谁，只要从后面推他一把，就可使其产生巨大飞跃。而leap也是Leaders Education and Achievement Program的缩写，意味“以精英教育与成功”。

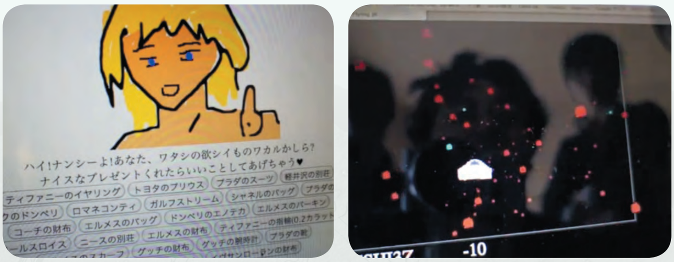
后来，我的这个想法得到了D2C公司（电通与NTT DOCOMO的合资子公司）的认可，我们约定由他们来提供启动资金。

在德克萨斯首府奥斯汀

咱们再把话题转到去美国出差时。GDC大会之后，我和辻秀美告别了其他成员，前往德克萨斯州的奥斯汀。在那里，我们将和公司的另外一些员工会合，参加South By Southwest（SXSW，西南偏南大会）这一美国最大的音乐、电影以及互联网盛会。

当时是井口尊仁先生邀请我们参加这个活动，我们才临时起行的。SXSW是极富盛名的大会，Twitter和Foursquare都是在这里一炮走红的。我们UEI公司也应该在这里好好表现一下。这次，我们大胆展示的是Zeptopad FOLIO产品，一个配备了强大的记录和编辑应用功能的电子画板。它的基础是NEC尚在开发中的LT-W，它有着双画面，使用安卓系统。这次参展时，Zeptopad FOLIO还处于原型机阶段。辻秀美自入职以来，就一直负责这款产品的开发策划和功能设计工作。这款产品，也彰显着我的信念，我要创造出新一代的“笔墨纸砚”（笔记工具）。

距SXSW大会开始还有少许时间，我乘坐游船，骑上两轮电动车，漫游于奥斯汀市。在



8小时内开发制作出的游戏：High & Low（左），3D游戏（右）

这段闲暇时光中，这个计划的整体构思渐渐明朗。

正当我沉浸在这新奇的构想中，激动得热血沸腾之时，在我深爱的祖国（日本），发生了一件百年不遇的事情。

悲　剧

傍晚，我正在旅馆房间中看电视，突然播放了一条爆炸性新闻。我紧盯着屏幕，不敢相信眼前的画面：日本发生了大地震，地震引起的海啸将日本东北地区毁灭殆尽！

我立刻给在美国当地的公司员工打电话，让他们赶快和远在日本的家人联系。当时，我自己也无法和家人取得联系。通过Gmail的聊天功能，我向公司下达了回家的命令。

然而，令人绝望的事情随后而至。核电站出现了危机！福岛的核电站一旦发生泄露，甚至连东京也将陷入危机。而我正远离祖国，一点儿办法也没有。无法保护深爱的人们，我对自己的无能为力感到极度的失望和痛苦。虽说在这种情况下，谁也没有办法。这就是所谓的绝望，真真切切。

第二天一早，井口先生把在当地的日本人都召集了起来，决定号召募捐。SXSW大会的筹备办事处也大力协助，在最显眼的地方设置了募捐场。募捐活动进展顺利，3天就筹到了2000万日元的善款。然后，我们通过奥斯汀当地的友人，把捐款托付给了红十字会。

我和家人也在第二天取得了联系。妻子的老家消失得无影无踪了，所幸，大家都平安无事。公司员工也没有人负伤。然而，对核辐射的担忧开始蔓延。核电站的糟糕状况完全不可预想，说不定什么时候核电站就会爆炸。面对这种状况，我也质疑起当时让在美国的员工都回日本的决定究竟是否正确。

地震后的东京

震后一个星期，我终于回到了东京。

面对灾后的惨状，如果再谈论游戏类的工作，有些不合时宜。我甚至一时提不起精神来工作。然而，身处灾区，深受地震影响的小舅子的一席话，让我的想法大为转变。

“公司没了，工作也没了。核电站的事情吧，我们也无能为力。唉，真是的，整天无所事事。我不能整天什么也不做啊。”

他们真的很想忘记震后的惨烈光景，仿佛在说：“就让我忘掉这些吧，谁能告诉我，现在我可以做些什么。”他们对生活的强烈渴望，直击我心。

我想我不仅能制作游戏供他们消磨时光，而且要把“制作游戏的过程本身，就是一种快乐的游戏”的这种妙不可言的美好感觉传递给大家。或许，这将是一个机会，让他们重新获得赖以生活的技能。

就在这时，9leap由构想变为现实的计划突然变得清晰起来。只要有电脑，无论谁，无论身处何地，利用HTML5，就可以进行游戏创作。而且，只要有智能手机，这些游戏就可以自由运行。而我坚信自己可以创造出这样一个世界：在这里，人人都可以创作游戏，领悟游戏的制作方法，由入门到精通，成为了不起的专业人才。想到这些，我下了决心，我要真正开始这项事业了。

ARC的诞生

从灾区回到东京，我便召开了临时董事会。眼前是一张张熟悉的面孔，阔别数日，我们再度重逢。我郑重入座，抛出了话题：“我想在秋叶原创建一个NASA。”

“什么？我们准备要进军航天事业了？”

“不是的。NASA象征着世界上知识和智慧的最高峰。大家知道吗？阿波罗登月计划的航空指挥官Gene Kranz当时只有35岁，然而在飞行计划方面，他的权力高于美国总统。只有35岁，这意味着什么？也就是说，在美利坚合众国，在拥有最高智慧的组织中，他们的现场指挥官，仅仅只有35岁！由此可见，为



SXSW上紧急开展的募捐T恤义卖。大部分人购买T恤的同时，又进行了捐款。



ARC的LOGO。上有古希腊语："不懂几何者不得入内。"来源于柏拉图创办的阿卡德米学园的门口竖牌。喻指ARC不只是一个事业部门，更是一个教育部门。

了实现开拓创新，要聚集优秀的人才，培养并且信任他们，而不应以年龄而带上有色眼镜。能力决定一切，这就是NASA了不起的地方，这也是美利坚合众国了不起的地方。"

"嗯，那我们准备做什么呢？"

"把人送往月球……呵呵，开玩笑啦。不过，我确实想到了一个宏大的计划：让困于软件星球上的我们，迈向一个新的阶段。"

"这个组织起好名字了吗？"

"ARC。"

"阿——克？"

"是的，秋叶原研究中心，简称ARC。集结起我们公司的最高智慧，打造世界上最先进的组织。"

"是像SONY的CSL那样吗？"

"不太一样。目前，我不准备在ARC里安排专业研究者。"

一个计划在我脑海里闪过，并逐渐明晰，首先我决心把ARC构建为一个没有专业研究者的组织。

"这个计划，我们不妨试试看。"

承担着公司最重销售任务的CTO水野拓宏发言了。他掌管UEI公司的主要业务。如果水野反对的话，我就只能花费大量心血去说服他，否则别无他法。

"不只是试试看，我很希望施行这个计划。但是，一年内必须要见到成效，这点请你和大家做出约定。"

从水野的立场上看，他很担心公司成员出现不满情绪。为什么要设置这个组织？如果没有做出明确的成果，那么维持这个组织的日常运行，会让公司员工认为是在牺牲自己的利益。

"好，那我们一言为定。"

就这样，2011年4月1日，秋叶原研究中心，简称ARC，作为公司的新组织，正式成立了。我兼任所长，成员以随公司一起成长起来的精锐员工为主体，伏见、高桥、近藤、辻等年轻员工也加入了进来。ARC的大总管是柿添先生，他曾在电通管理过风险投资，经验丰富。

enchant.js项目和9leap项目都统统纳入了ARC的管理范围。没过多久，4月17日，enchant.js便正式发布了，而9leap项目也于5月1日启动。开始后仅仅10天，便收集到了100件作品。9leap项目，真如火箭升空一般，气势恢弘地迅速展开了。

我所偏爱的键盘图鉴

文 / 创梦株式会社　滨野圣人
khiker.mail@gmail.com
Twitter：@khiker
译 / 卫昊

第3回

人体工学键盘的最强者

KINESIS Contoured & Maltron dual hand 3D



图片1　KINESIS Contoured 键盘（Advantage pro）

引子

本期将介绍可变更键盘布局的可编程键盘的代表——KINESIS Contoured 键盘。此外，还将介绍一款非可编程键盘——与 KINESIS Contoured 外形相似的 Maltron dual hand 3D 键盘。

KINESIS Contoured 键盘

KINESIS Contoured 键盘是美国 KINESIS 公司旗下著名的人体工学键盘。目前，在售的 USB 接口的 Advantage 系列中，除普通版本之外，还存在其他多个版本，如专业版（图片 1）、使用 Dvorak 布局印刷键帽的版本、采用 linear feel 按键轴的版本（图片 2）等。

专业版除了多项附加功能外，还附赠一个拥有一个开关的脚踩踏板（图片 3）。也可以另外选购拥有三个开关的脚踩踏板（图片 4）。

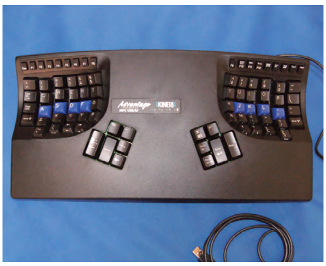
特点

KINESIS Contoured 键盘的特点大致如下：

- 可以有效使用大拇指的碗状外形。
- 方格形的布局。
- 可编程功能。
- 采用机械轴。

碗状的外形以及方格形的布局，可以使手腕和手指在几乎不需要移动的情况下按键。同时，相较于一般的键盘，大拇指可以敲击的按键增多，所以可以更有效地使用大拇指。

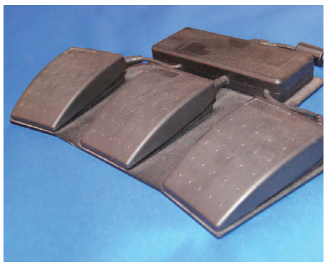
可编程功能允许使用键盘本身的设置功能，以自由修改任意按键的位置。在操作系统中，只需设置成英文布局即可，不需要其他的任何设置。按键排列的变更本身并不需要特别的软件。首先保持键盘接入 PC 的状态，按住 program 键 + F12 进入映射模式。按顺序输入复制前的按键、复制后的按键，再一次按住 program 键 + F12 即可完成映射。并且，还可以定义一个一对多的按键宏。这个设置也不需要特别的



图片2　KINESIS Contoured 键盘（Advantage）



图片3　Contoured 专用脚踩踏板（单键）



图片4　Contoured 专用脚踩踏板（三键）

软件。专业版还可以锁定重新映射过的布局，让第三者无法对其进行变更。

键轴的部分采用的是Cherry的机械茶轴。Cherry的键轴有很好的耐用性，以及良好的击键感。茶轴的特点是按键在按下之前有少许阻隔感。linear feel版本使用的是Cherry的红轴。红轴与茶轴不同，没有了按键按下之前的阻隔感，特点是随着按键按下的深度增加，按键的反弹会随之线性地加强。

购买方法

KINESIS的产品在日本有代理店[①]。比从KINESIS公司直接购买要更便宜，并且一年中会有多次优惠活动，可以以便宜好几个百分比的价格购买到。也可以通过Amazon购买。以上两种购买方式，价格大约都在3万日元左右（折合人民币约1776元）。

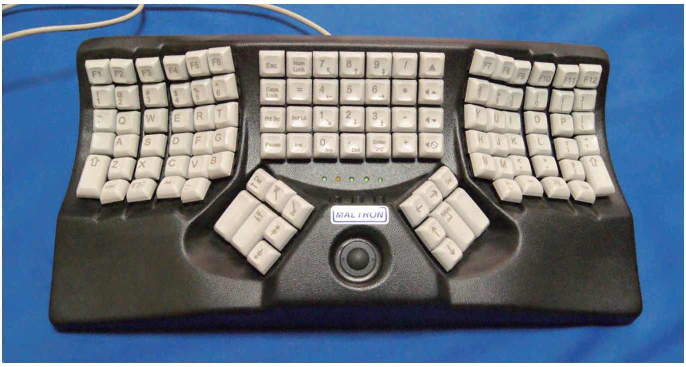
Maltron dual hand 3D键盘

Maltron dual hand 3D 键盘是Maltron公司[②]的老牌人体工学键盘（图片5）。有采用Dvorak布局版，配备轨迹球版、无轨迹球版等多种类型。

特点

Maltron与KINESIS外形十分相似，其区别如下（图片6）：

- 功能键为普通材质，不是橡胶制成。



图片5　Maltron dual hand 3D键盘

- 小键盘区独立存在于中央。
- 键盘布局有些许差异。
- 配备了轨迹球。
- 没有可编程功能。
- 大拇指的按键区域在凹陷下去的位置。

Maltron的轨迹球设计较小，拉远了和键盘Home Position（基准位）的距离。此外大拇指按键区域的凹陷，和KINESIS相比也使按键距离变远。整体感觉像是为手大人群设计的。

除此以外，Maltron也采用了Cherry的机械轴，这里使用的是黑轴。黑轴基本上和红轴类似，特点是按键的反弹较强一些。

购买方法

在日本虽然有Maltron的代理店，但价格却相当高，约12万日元（折合人民币约7119元）。官网直购商店价格为435欧元，但不知道是否可以送货到日本。另外，ebay上偶尔也有售卖。笔者就是从ebay购买的，最终竞标价约为500美元。

◆ ◆ ◆

本期介绍了两款编程键盘。Maltron更适合手大的人群，但两者都是优秀的著名人体工学键盘，如果你被腱鞘炎困扰的话，这两款键盘是值得购买的。本期虽然没有详细介绍Cherry机械轴，但Cherry机械轴也是关乎键盘性能的一大特点。有关Cherry机械轴将在今后介绍。



图片6　Maltron和KINESIS的对比

① http://edikun.ocnk.net/。中国读者可以选择到Amazon购买或找相应渠道代购。——译者注

② http://www.maltron.com/

创客在行动

2013旧金山湾区制汇节见闻

文/坪井义浩 （TSUBOI Yoshihiro　ytsuboi@gmail.com　Twitter @ytsuboi） 译/苏祎
协助：（株）switch science　http://www.switch-science.com/

旧金山湾区制汇节

继上个月参加深圳制汇节后，此次我们又去参加了世界最大规模的旧金山湾区制汇节（Maker Faire Bay Area）。

正如其名，该制汇节在美国硅谷地区的San Mateo举行。San Mateo距离旧金山机场非常近，打车费用在30美元之内。

笔者在switch science的展台出展，所以未能深度体验整个制汇节，但也算是简单转了一圈，在此向大家介绍一些有意思的内容。

▼图片1　食物展台的烟熏机



与日本的制汇节相似的是，来客也多是以家庭为单位，使得制汇节更像是面向家庭的活动。制汇节的英文使用了“Faire”而非“Fair”，这源于中世纪的节庆活动圣何塞Fantasy Faire，如其所指，确实是盛大的节庆活动。制汇节上也有很多食品售卖展台，尺寸在日本人看来是很大的。例如其他出展者推荐笔者尝试的烤火鸡腿烟熏机器，规模如图片1所示。

当然，很多展品尺寸在日本无法想象。比如图片2中所示巨大的Lifesize Mousetrap，类似于“毕达哥拉斯开关”[①]，这般庞然大物真不知是如何搬运到会场中的。另外，还有小型潜水艇的展出（图片3）。

小创客

最近美国的创客策划了越来越多面向小

① NHK教育台的一个培养小孩子思考方法的电视节目。——译者注

▼图片2　Lifesize Mousetrap



▼图片3　潜水艇



朋友的活动。例如目前正在实施的“小创客”(Young Makers)项目[②]，它以通过气压使棉花糖飞起的Marshmallow Cannon声名鹊起，甚至有新闻称美国总统奥巴马也将邀请该少年创客到白宫一起放飞棉花糖。若有兴趣的话，可以通过搜索“Extreme Marshmallow Cannon”来了解具体情况。

当然，可能大多小创客的成年监护人是工程师，并且也帮了他们不少，但也有一些作品即使从成人的角度来看，也是非常有意思的。例如，能让人联想到Blue Man的乐器，使用PVC管制作乐器(图片4)，使用了Makita的电动工具作为动力的气垫船(图片5)，巨大的柏青哥(图片6)等。这些也都是在日本城市中心的住宅条件下，基本无法进行尝试的大制作。在美国，创客受到关注的原因之一是制造业层面上的回归，而在日本，制造业日渐没落的原因，也可能源于类似的活动和玩乐机会的缺失。

机器人

笔者在深圳时深切体会到，在国外比在日本更容易遇上机器人制作爱好者。美国的极客当中，星球大战相关内容依然是主流。现场有一处聚集了大量星战中的机器人，很多人都在开心地拍着照片(图片7)。这些机器人的展示方式也非常巧妙，大家可以在星战宇宙船的巨大海报前为其拍照。与东京的制汇节相比，这样的展示方式体验更好。另外，与此类似的作品还有BEER2D2(图片8)。BEER2D2看上去是使用啤酒桶制作的机器人，它可以在会场中到处跑动。

在日本，有机器人方面的竞赛，而在美

② http://youngmakers.org/

▼图片4　PVC管乐器



▼图片6　巨大柏青哥



▼图片8　BEER2D2



▼图片5　气垫船



▼图片7　R2 BUILDERS



▼图片9　竞赛机器人



国，也有展出用于竞赛的机器人（图片9）。可以投出飞盘的机器人吸引了笔者的眼球，让我感慨不已。它不仅可以用飞盘击中目标，还可以将掉下来的飞盘回收起来。它的自律性让我感触颇深。

Cosplay

会场中也有很多穿奇装异服的人，让不了解创客文化的美国人惊叹“好多奇怪的人”（图片10）。有些人更是拿出了看家本领，费尽心思，自己做了模具并用树脂制做部件（图片11）。

另外，也有人穿着游戏中的带刺布偶装。而这些尖刺貌似还会随着身体的运动发光（图片12）。另外，也有扮演成恶魔的人（图片13），这套衣服的独到之处是配备了声音转化器，嘴的部分可以根据声音改变点亮模式。

在这里还遇上了之前在深圳遇到的人，她的名字叫Katia Vega[③]，研究方向是可穿戴电脑技术（图片14）。她的眼线是用导电材料制作的，所以通过眨眼即可对一些设备进行控制，指甲上的RFID（天线射频识别），可以对手指的触碰进行识别。

hackerspace

笔者在各地走访过的hackerspace，也在制汇节出展介绍自己的空间。这次在湾区，我对bio curious（Bio系中罕见的hackerspace）很感兴趣，所以特地去看了下展区。如之前在Make: Japan的博客中介绍的，他们制作出了DIY Bio Printer（图片15）。最近，3D打印机的话题一直很热，而在技术阵地最前沿，也在研究着生物打印技术，该技术可以将细胞进行3D打印，并制作成用于移植的组织器官。

该打印机是通过使用喷墨打印机的墨盒，将大肠杆菌印刷到培养皿的琼脂上。当然这还算不上最前沿的技术，但通过DIY轻松地尝试类似领域的技术，确是件非常新潮的事情。这个打印机的制作方法公开在名为instructables的网站上[④]。在instructables网站上，还有很多有意思的项目。

③ http://katiavega.com/

④ http://www.instructables.com/id/DIY-BioPrinter/

▼图片10　专业的Cosplay



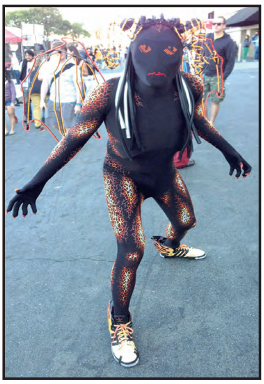
▼图片11　武器的模具



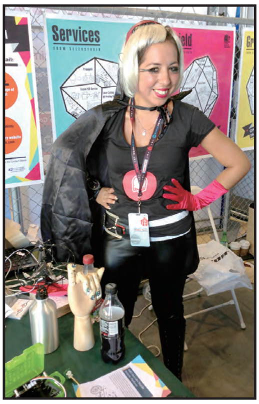
▼图片12　满身是刺的人



▼图片13　恶魔造型的人



▼图片14　Katia小姐



日本的出展者

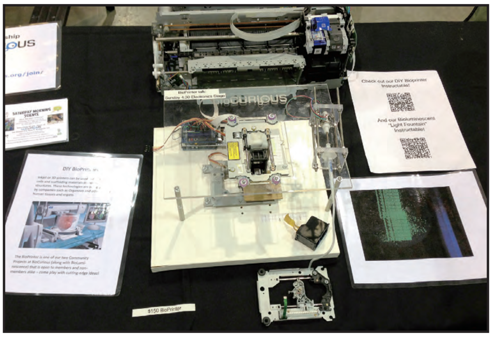
此次湾区制汇节也有多名日本的出展者参加，在此介绍一下。

首先在湾区居住多年的Kazuhi氏，他的展品名为GlueMotor（照片16）。Kazuhi氏在去年的湾区制汇节出展，并且当选了Editor’s Choice，在Make杂志中还发表了文章。而今年，他荣获了Education Award，再次桂冠加身。他是位非常有亲和力的大叔，他还带笔者去逛了湾区的旧货店。

GlueMotor基本没有使用电路部件，而是通过iPhone等智能手机的耳机端口，控制伺服的内容。GlueMotor制作非常简单，主要部分为编写智能手机的应用，对于本杂志的读者来说应该是易如反掌。这个网站上[5]有他编写的相关信息，若有兴趣，大家不妨访问查看。

另一位日本的出展者是伊藤氏（@itog）。他的展品为FourBeat（图片17），是Android系统下很多应用可以利用的一系列按键。被称为日本版Kick Starter的CAMPFIRE，为这个作品建立了融资基金，很多小朋友都聚集在展台并狂热地连击着按键。

▼图片15　DIY BioPrinter



同样来自日本的出展者新里氏（@hirotakester）展出的是，可以与心跳相关联的发光史莱姆（图片18）。这里也聚集了很多小朋友在玩史莱姆。

总结

之前提到的Kick Starter[6]网络服务，是近期的热门话题。此次的制汇节中，也有众多出展者在宣传着自己在Kick Starter融资的内容，希望通过筹集资金进一步推进自己的项目。此次英特尔、丰田、雪佛兰、福特等大企业赞助制汇节的举动，也让人感受到了创客在社会中的影响力。

⑤ http://www.gluemotor.com/

⑥ http://kickstarter.com/

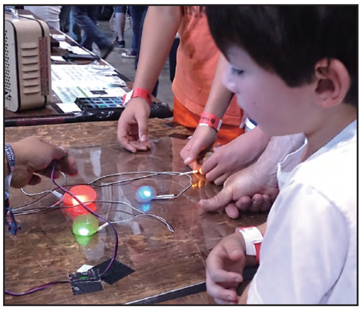
▼图片16　GlueMotor



▼图片17　FourBeat



▼图片18　新里氏的史莱姆



特辑1 让我们重新审视数据的价值

学习数据分析，从这里开始

— Excel・R・Mahout・大数据 —

“大数据”给人一种很阳春白雪的印象，容易让人以为它是一个与自己无关的数据分析领域的术语。然而，在当今这个时代，为了在激烈的竞争中求生，不可欠缺的就是创造新机遇和挖掘新需求。因此，无论面对的数据量是大是小，我们都需要数据分析技术，从而获取有用信息，这样的需求与日俱增。

然而，分析数据及正确预测，都离不开数学和统计学知识。是知难而退，还是迎难而上，就成为决定软件工程师是否具有强大竞争力的分水岭。

本特辑面向软件工程师，详细讲解日趋重要的数据分析技术。同时，我们还介绍有助于掌握必要知识的学习方法及参考书，以及通过使用Excel、R、Mahout等工具来掌握数据分析的手法。

CHAPTER 1

献给软件工程师们

数据科学入门及学习指导

“我对机器学习或者数据分析有兴趣，但是相关的专业知识纷繁复杂，不知道从哪里开始着手。”想必不少朋友有这种苦恼吧。本文面向软件工程师，以介绍数据科学的兴起和概要开端，逐步讲解数据科学的必要知识与技术，同时也给大家提供必要的学习指导。

文/柏野雄太　BakFoo株式会社董事长　译/雷军

背景——从大数据热潮开始

近来，大数据热度稍有减退，但此前大家对大数据的追捧几近狂热。从不了解情况的商家，到连篇累牍的经济类报纸，大家对大数据都进行过大肆宣扬。在《日本经济新闻》上，甚至都出现过Hadoop、MapReduce等字眼。但是此后两年间，情况并没有因大数据而出现大的变化。耐不住性子的人便开始感叹：“大数据不行了！[①]有人甚至开始撰写具有煽动性的报道：“显然不是什么大数据，却还要伪装，简直就是拿钱打水漂！[②]

然而，两年之后，大数据不但没有消亡，反而越来越让人们坚信，今后的数据将会越来越大，大数据相关的技术和知识将会变得越来越重要。在现存机构的数据之上，通过网络和客户进行数据交互而引起的数据增加，开放资料数据的增加，社交网络数据的激增，以及为了简便地读取这些数据而层出不穷的新手段，所有这些都表明数据规模正变得越来越大。

5年前有人如此评价云计算：“只是炒作而已。”当时，这样想的人很多。但是现在，云计算显然已经完全主导了服务器端的计算机了。

和云计算一样，大数据也改变了计算机原有的使用方式。无论是谁，最终都将不得不顺应这一巨大潮流。差别仅仅在于实现这种转移需要的时间长短，是几年还是几十年而已。无论你喜欢与否，今后若没有处理大数据知识和技术，就无法使用计算机，这样的时代即将来临。正因如此，软件工程师们必须掌握大数据的处理分析，否则接下来的职业生涯将变得异常艰难。

对于软件工程师来说，最关键的问题是掌握必要的知识和方法，然而，这是一个庞大的知识体系，因此，学习起来不会像学习网络构架那样轻松。

数据科学

数据科学是一个专业术语，指与大数据处理相关的知识体系和技术系统[③]。实际上，数

① Big Data is dead（http://venturebeat.com/2013/02/22/big-data-is-dead-wahts-next/）。
② Most data isn' t 'big' and businesses are wasting money pretending it is（http://qz.com/8166/most-data-isnt-big-and-businesses-are-wasting-money-pretending-it-is/）。
③ 最早阐述数据科学的是O' Reilly的Mike Loukides先生的What is data science?（http://radar.oreilly.com/2010/06/what-si-data-science.html）。

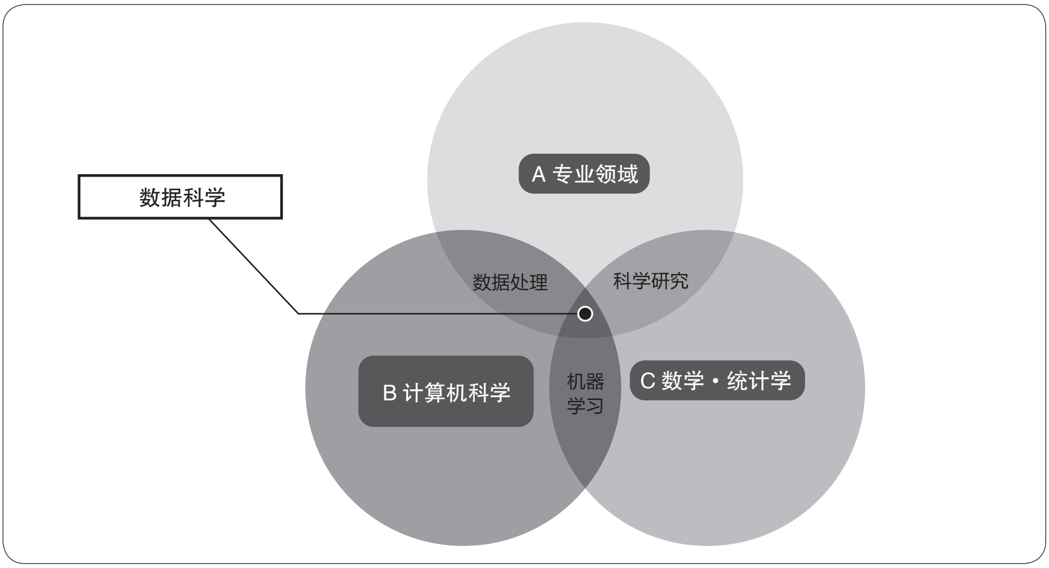
据科学是统计学/数据分析、软件编程技术/计算机科学知识、实际商务及研究领域等的专业知识的紧密结合。从事数据科学工作的人被称为数据科学专家。数据科学经常被称为专业知识、数学知识以及计算机科学知识的综合体。数据科学的知识体系如图1所示。④

数据科学包括三大基本领域：商务以及研究的专业知识或技能（A领域）、软件编程技术及计算机科学的基本知识（B领域）、数学以及统计学知识（C领域）。这三部分之间互有重叠，例如，A和B的重叠部分被称为进行商务和研究的数据处理领域，B和C的重叠部分就是最近很盛行的机器学习领域，A和C的重叠部分，很久以前就已经被称为科学研究领域。数据科学就是A、B和C这三大部分都重叠的中心部分。笔者坚信，这部分将来必定会成为软件工程师专门研究的领域。

在数据科学的发祥地美国，一位任职于Google的著名微观经济学家哈尔·范里安，7年前曾经说过："统计是未来迷人的工作。"⑤由此开始，广义上的统计学⑥即数据科学引起了IT界的广泛关注。

然而，把数据科学作为独立术语使用的做法也遭到了一些批判。⑦批评者们认为，如同CERN（欧洲原子核研究机构）为了发现希格斯玻色子粒子而进行的实验一样，⑧将理论和

▼ 图1　数据科学的知识体系



④ 这幅图是根据IA Ventures公司的著名数据科学专家康威先生的数据科学定义绘制的。最近他在《数据科学通信园地》发表的文章引发了热议。

⑤ http://www.nytimes.com/2009/08/06/technology/06stats.html

⑥ 这里虽然用了“统计学”这个词，但是Google的范里安先生的主张是，传统的统计学主流派的频率主义与其说是奈曼–皮尔逊理论的统计学，不如说是以机器学习为主流的贝叶斯统计学。

⑦ Moshe Y.Vardi “Science has only two legs” http://delivery.acm.org/10.1145/1820000/1810892/p5-vardi.html?ip=110.5.47.50&acc=OPEN&key=1B55DF923F77674F55057ED4F3766CA0&CFID=217973290&CFTOKEN=96019330&_acm=1368853946_dc072110a74bb3bb1259ba9527e3d119

⑧ CERN的大型强子对撞机（LHC）试验开始后便会产生40太字节/秒的数据。为了对这些数据进行获取、过滤、存储和分析，必须有新的“科学”以及“专家”。

实践相结合的科学研究，需要利用大规模计算机来执行。因为这样的科学研究，无非是用现代的计算设备代替以前的人工操作而已，所以不能认为是新兴的科学领域。

但是，当进行理论与实践研究需要处理的数据变得特别庞大，科学家们已经不能顺手处理时，新兴的专门从事这种工作的专业人员就变得必不可少了，这些专业人员理应被称为数据科学专家。

实际上，2010年数据科学这个术语刚刚出现时，遭到了很多批判。但是到了2013年，专家团体已经逐渐形成，每个月在各地都会举行数百人规模的会议。并且，从劳动力市场角度来看，数据科学专家的需求也已形成一定规模，这样一来，当初的质疑之声便销声匿迹了。

数据科学的构成要素

对初学者来说，图1所示的三个领域之中，除了领域A的专业知识以外，统计学/数据分析和计算机科学的难度都很高，不易掌握。

数据科学专家必备的统计学知识是描述统计、推断、检验、回归等基础知识，以及被称为数据科学主流统计学的贝叶斯统计学。数据科学家所必备的计算机科学知识包括：对数据结构的理解力；熟练运用多个分布式计算系统环境的能力；[9]学习多种编程语言的能力；[10]对数据分析框架的组合能力[11]等。

机器学习理应被称为统计学和计算机科学的嫡子，是数据科学的核心知识。机器学习的字面解释是，为了处理某个任务，机器（软件/程序和硬件）通过学习过去的经验，获得高效处理该任务的各种方法。在网络教学网站Coursera上，吴恩达博士[12]有个著名的讲座，其中引用了汤姆·米切尔先生关于机器学习的明确定义："假设把利用成绩P实现任务T所能获得的经验E作为前提条件，如果有某个程序（机器）可以利用现有的经验E来更好地完成任务T，那么，就可以称这个程序（机器）具有学习能力。"

对于初学者来说，学习机器学习，的确是特别困难。它不但是日新月异的领域，而且也不像传统物理学那样，是可以从第一个原理开始推导的知识体系，它是一个庞大的探索式思维的集合。

总之，学习数据科学是要经历一个痛苦的过程的，但这个过程中也是有技巧的。

学习技巧及资源介绍

前面已多次阐明，数据科学要求的基础知识量相当多，所以学习它是比较困难的。学习如此大量的知识，最好的方法当然是，在大学或研究生院充分利用讲义、练习、讨论、自习的机会，以及花时间认真踏实地彻底理解教材、做练习、编写程序。

然而，上班族要像全职学生一样用全部的时间去学习是不现实的。这里最想推荐给上

⑨ 具体指Hadoop、Spark、Storm、Omq、Cassandra等。
⑩ 具体指Java、C/C++、Python、R、Matlab、Octave、Scala等数据分析的常用语言。
⑪ 具体指scilit-learn、Mahout等主要构架，包括各种构架的有mloss.org的软件分类（http://mloss.org/software/）。
⑫ Andrew. NG. 美国斯坦福大学计算机科学系副教授，人工智能实验室主任，在线教育平台Coursera联合创始人。

班族的是，利用近年来迅速发展的网络教育（Massive Open Online Course，简称MOOC）⑬，网络教育可以实现在美国境外以最小的投入，接受全美最高学府的课程教授。当然，所有的教材都是英文的，但并不难懂。软件工程师不是依靠英文为生的，只要能听懂和读懂相关专业的英文教材就足够了。⑭

另外，本章将省略书籍介绍。虽然好书很多，但想在短时间内通过自学掌握大量知识，仅依靠书籍还是相当困难的。

在此还将省略作为数据科学专家所应必备的计算机周边知识、主流计算机语言、数据分析/机器学习的框架、最优算法的研究等计算机科学知识。本章将重点介绍统计学及机器学习的MOOC系列。

那么，赶快开始学习利用MOOC的技巧吧。

● 下载视频以便在任何时间能够收看

如果选择Coursera，推荐使用Python编写的工具coursera-downloader⑮。如果选择YouTube，可以使用Firefox等下载扩展功能丰富的浏览器。如果还是不能下载，可以使用一些屏幕录像软件记录视频讲义⑯，以便在任何时间都能够收看。一般情况下，Coursera课程每次大约只有15分钟。利用零碎的15分钟，集中精力学习应该是比较容易实现的。

● 利用平板电脑终端在任何地方都可以收看讲座

笔者就是使用Nexus 7，无论是访问客户时，还是泡澡的间隙都可以收看讲座。短时间段收看时，最理想的是能自动保存播放位置。Android的MX动画就具备这种实用功能。

● 动手做练习加深理解

即便跳跃式地收看讲座也没有关系，根据笔者的经验，关键是要亲自动手做练习才能加深理解。为了解决一道练习题，有时候需要反复多次收看一个讲座，才能理解到底结果是怎样得到的，这个过程自然就会加深对知识的理解，因此，手脑并用才能把知识巩固在自己的大脑里。

● 不必追求完美

就软件工程师而言，学习的目的是在短时间内学会一些基础知识，不是为了研究，所以不必达到非常完美。

下面将具体介绍MOOC讲座，以及一些能用日语学习的网站。

描述统计学/推断/检验/回归

这些都是统计学的基本术语，非常重要且十分枯燥。快速吸收这些知识的好办法就是使用MOOC。edx的Stat2.1x、Stat2.2x、Stat2.3x的讲座简明扼要、内容充实，一周内就能掌握概率变量、期望值、分布、贝叶斯推断规则、推断等基本概念。

https://www.edx.org./course/uc-berkeley/stat2-1x/introduction-statistics/594
https://www.edx.org./course/uc-berkeley/stat2-2x/

⑬ http://en.wikipedia.org/wiki/Massive_open_online_course
⑭ 很遗憾，到2013年5月为止，关于数据科学的日本语的在线讲座一个都没有。
⑮ https://github.com/siddharthasahu/coursera-downloader
⑯ 利用搜索引擎检索关键字“screen recording software open source”等就能找到开源的屏幕录像功能的软件工具。但是，有的网站只能允许流媒体收看，请仔细确认说明。

introduction-statistics/685
https://www.edx.org./course/uc-berkeley/stat2-3x/introduction-statistics/825

如果不用MOOC，那么想利用R语言学习统计学基础知识的最佳日语资源，就是群马大学社会情报系的青木繁伸先生的网站。其中的《统计学自学笔记》是学习统计学的基础知识以及R语言的绝佳资料。

http://aoki2.si.gunma-u.ac.jp/lecture/index.html

另外，筑波大学医学医疗研究系的冈田昌先生创办的RjpWiki，也是对R语言及相关内容有定期介绍的日语情报网站。

http://www.okada.jp.org/RWiki/

贝叶斯统计入门

没有贝叶斯统计知识，就不能说是完整的数据科学。适合入门学习的首选讲义是，在PyCon US上的Bayesian statistics made simple视频讲座。该讲座配有独特的贝叶斯统计的代码库，虽然刚开始稍微有些抽象，但是在动手编写简单Python代码的过程中，就会渐渐地理解贝叶斯统计的基础知识，大约2小时就能掌握了。

http://pyvideo.org/video/1724/bayesian-statistics-made-simple

数据分析入门

想要尽快入门数据分析，Coursera的Computing for Data Analysis，是利用R语言进行数据分析的最佳入门讲座。

http://www.coursera.org/course/compdata

数据科学入门

现在Coursera的课程中，直言不讳地说，关于数据科学入门讲座Introduction to Data Science，虽然内容实用，从关系型数据库、Hadoop、MapReduce到数据整理的方法都有全面介绍，但是遗憾的是，由于数据科学的知识范畴广大，也使得入门学习讲义像百科词典一样杂乱。讲义和课题的内容偏离较多，那么，对于本来就没有太多时间的上班族来说，学习的技巧就是抓住关键的要点。

https://class.coursera.org/datasci-001/class/index

机器学习入门

MOOC的Coursera的联合创始人、世界级飞行机器人研究者吴恩达博士的讲座Machine Learning，知识水准高，表达简洁明快，对于初学者来说，非常合适。配合使用Octave动手编程练习，收看这个讲座，就能够掌握机器学习的基本知识。该讲座适用于任何水平的初学者。

https://class.coursera.org/ml-2012-002/

加州理工大学的亚瑟 · S. 阿布 - 穆斯塔法博士的讲座也十分优秀。然而，按讲座顺序逐一收看，直接选择感兴趣的专题讲座效果更好。

http://work.caltech.edu/library/index.html

MOOC以外，与机器学习有关的日语网站中，内容最充实的是产业技术综合研究所神岛敏弘先生管理的“朱鹭之社Wiki”。

http://ibisforest.org/index.php?%E6%A9%96%A2%B0%E5%AD%A6%E7%BF%92

还有，技术评论社的网站（gihyo.jp）的“开始机器学习吧”，是面向初学者的连载入门讲座，课程亲切周到，配有Python代码的讲解有助于理解，是一个非常好的学习资源。

http://gihyo.jp/dev/serial/01/machine-learning

机器学习的实践

作为机器学习的实践讲座，如果要专门使用scikit-learn，那么必须收看PyCon US的讲座。scikit-learn开发中心的Jake Vanderplas先生和Oliver Grisel先生的培训课，易懂易学，同时提供了要点提示丰富的笔记，非常有助于学习，值得推荐。

http://pyvideo.org/video/1665/an-introduction-to-scikit-learn-machine-learning
http://pyvideo.org/video/1719/advanced-machine-leraning-with-scikit-learn

机器学习高级篇

机器学习的高级水平讲座中，推荐Coursera的 Neural Networks for Machine Learning。该讲座是机器学习领域中流行的高级水准的讲座。

http://class.coursera.org/neuralnets-2012-001/class/index

而且，在videolectures.net中，收录有世界级计算机科学系学会的辅导培训的压缩版资料，还有著名的克里斯托佛 · M. 贝肖普先生的“模式识别与机器学习”的部分讲义[17]，该讲义最终通过了博士论文答辩！一定要充分利用这样的资源。关于videolectures.com资源，因为篇幅有限这次不能全部介绍，以后有机会再详细介绍。

http://videolectures.net/

最后

至此，我概略地介绍了使用MOOC学习数据科学的学习资源。实际上，除了这些内容之外，数据科学的基础知识和技术还有社交网络分析，地理空间的数据分析以及市场的数据分析等应用。机器学习的基础知识还有自然语言处理、图像识别、自动翻译、语音识别以及交通工具的自动运行等应用。

数据科学知识包罗万象，因篇幅有限，很多相关的知识未能介绍，也省略了不少个人性的意见[18]。只能就此罢笔，非常感谢你的耐心阅读。

---

⑰ http://videolectures.net/mlss04_bishop_gmvm/

⑱ 数据科学是一个大领域，为了能够处理非正规数据，需要对数据进行数据整理，因此需要有分散型计算机处理环境及时保存大数据，并在必要时迅速检索。关于这些内容，有机会再与大家交流。

特辑1 学习数据分析，从这里开始

CHAPTER 2-1

数据分析入门必备知识

让数据分析工具助你一臂之力 [Excel篇]

非数据分析专职人士，一般很难对数据分析究竟要分析什么，怎么分析有具体概念。关于如何分析，我们首先用日常熟悉的Excel数据分析来实际尝试一下。

文/高木基成　三得利系统技术株式会社　译/杨驹武

前言

不知有多少读者对“数据分析”有具体的概念，如果你具备相关知识和经验的话，就没有必要在这一章上花太多时间。但对于像笔者这样，学生时期、工作阶段都没有接触过数据分析的读者来说，本文也许可以提供一些有趣的信息。让我们稍微花一点儿时间，一起来学习一下分析数据的方法。

数据分析的现状

在IT界，大数据、数据科学、机械学习、统计分析等术语现在满天飞。人们希望从大量产生的数据中，提取有价值信息并加以利用，因此，数据分析备受关注。另外，具有数学和统计学专业知识的数据科学家的数据专家也备受关注，现在业界也正在议论着数据分析方面的人才不足和人才培育的问题。

在普通企业内部，有很多潜在的基础数据分析需求，可以说正处于摸索合理数据分析机制的阶段。对于数据科学家、业务的一线负责人、信息系统部门等部门的负责人，数据应该由谁来分析，这个问题争论不休。这种状况下，IT工程师该如何面对数据分析呢？

什么是数据分析

首先，要明确什么是分析？字典中是这样定义的：“把一件事物、一种现象、一个概念分成较简单的组成部分，找出这些部分的本质属性和彼此之间的关系。”数据分析即是为了阐发现实中发生的现象，而从数据中提取有效信息的分析。

数据分析的分类

数据分析大体上可以分为两类：发现型和验证型。发现型分析不预设目的，对数据进行多角度分析，以发现新信息为目的。这种方法可有效获取新信息，受到了极大的关注。但是，实际工作中的数据分析基本上都是验证型。验证型分析的目的是提供证据来证明预想的因果关系。

分析的客体

在实际工作中进行数据分析时，需要提供

对企业的目标有价值的信息。我们来简单介绍一下具体的分析需求。

- 销售额/需求供给预测
- 销售额因素分析(怎样才能畅销)
- 顾客分析(什么样的顾客买了什么商品)
- 提高工作效率

从这些例子我们可以发现，通过数据分析可以知道两件事情：①通过分析过去来把握现状；②通过分析过去来预测未来。若没有数据分析，就只能靠经验和直觉来开展业务。比如，常常会实施一些类似于“因为新商品的顾客多是十几岁的人，所以面向年轻人展开促销活动”、“因为电视广告最有助于提高销售额，所以延长播出时间”的并非基于客观信息的经营策略。在商务领域，经验和直觉无疑是强大的武器，但经验和直觉因人而异，稳定性不强。数据分析可以把无形的信息形式化，这样可以促进业务稳步推进。

数据分析的流程

数据分析的基本流程如表1所示。如果想要预测未来，就要制作分析阶段④的模型。如果只想把握现状，就没有必要制作模型。

表1的实施步骤中，大家认为哪一阶段最难？大家是否感觉分析数据和制作模型比较难呢？因为它们需要专业技能。当然，每个阶段的难度因分析的内容和项目而异，不能一概而论。但是我访问了一些有数据分析经验的人，得到的大多数回答都是“相比起来，收集整理数据花的劳力和时间是分析工作的好几倍”。比如，想要对其他公司的销售额进行比较分析的话，其他公司的销售额信息并不容易获得。而且经过他人加工过的信息可信吗？稍微想一下就能知道，这里面肯定有很多问题。

IT工程师与数据分析

通过前面的内容，大家对数据分析应该有了更深的了解。在大数据时代，各式各样的信息被存储到系统中，尽管大家都希望能使用数据分析，但现在连能进行基础分析的企业还很少。明知道数据大山里藏着宝，却放置在那里不管不顾，着实可惜呀。

IT工程师该如何对待数据分析呢？笔者本

▼ 表1　数据分析的大致流程

分析阶段	实施项目	实施内容
①	设定目标	・确定通过分析想要弄明白哪些内容 ・明确基于分析结果想要采取的行动
②	收集整理数据	・汇集分析所要的数据 ・清除异常数据，统一数据的精度
③	分析数据	・使用分析方法分析数据 ・在必要时返回上一阶段
④	制作模型	・按照分析结果构建数学模型 ・验证数学模型的准确性
⑤	应用到业务中	・根据分析结果和预测结果确定要实施的行动 ・预测业务中发生的变化，进行更深入的分析

人也没有明确答案。也许有人会想就让业务负责人或者数据工程师去干吧。但是我想我们应该更积极地去对待。因为数据由系统生成并保存，所以，由开发或者运用系统的IT工程师，来掌握数据分析的技能进而推进数据的使用也是顺理成章的。

整理数据

现在我们开始动手操作，第一步我们先使用Excel。尽管大数据埋藏无尽珍宝，但是分析少量数据也可以获得一些有用信息。虽然距离使用大数据还很遥远，要挑战的课题还很多，但作为练习是可以的。

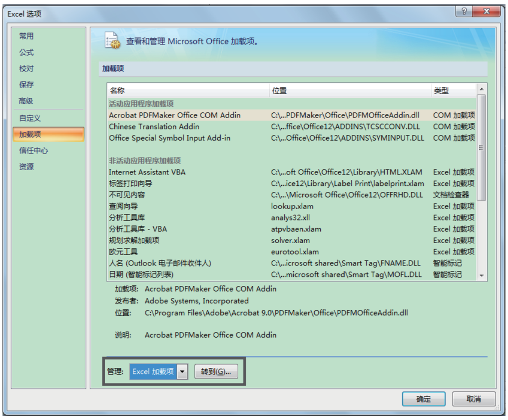
现在我们来对照表1的流程对数据进行整理，这属于分析阶段②收集整理数据的相关工作。

安装分析工具库

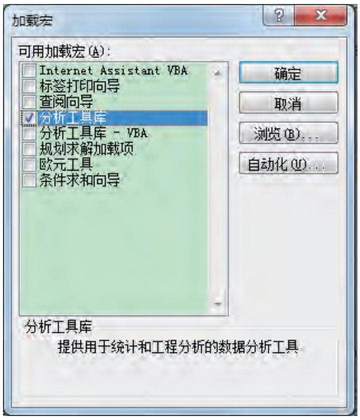
我们首先安装Excel的分析工具库。不同的Excel版本安装方法多少有点区别。基本上都是从加载宏选择框中选择分析工具库。微软的官方网站上有安装步骤，请见链接1。

例如Excel 2007，从“Office按钮”选择“Excel选项”，在打开的的窗口的左边选择“加载项”（图1），然后在加载列表下面的“管理”下拉框中选择“Excel加载项”，按下“转到”按钮，会打开用于选择可用加载宏的对话框（图2）。在“可用加载宏”框中，选中“分析工具库”复选框，然后单击“确定”。就开始了加载宏的安装。安装完成后，“数据”菜单里应该会加入“分析工具库”（图3）。

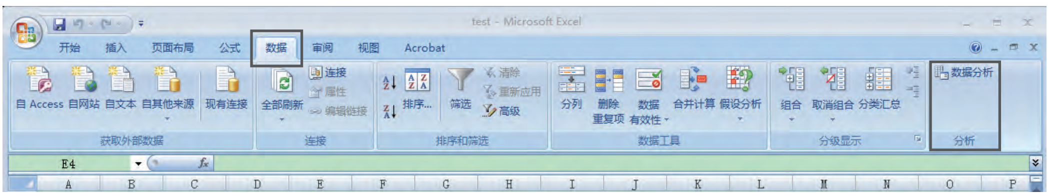
▼ 图1　Excel的选择界面



▼ 图2　可用加载宏的选择界面



▼ 图3　分析工具库被添加到工具栏上



▼ 链接1　添加分析工具库的步骤

版本	URL
2003	日语：http://office.microsoft.com/ja-jp/excel-help/HP001127724.aspx 中文：http://office.microsoft.com/zh-cn/excel-help/HP005203873.aspx?CTT=1
2007	日语：http://office.microsoft.com/ja-jp/excel-help/HP010021569.aspx?CTT=1 中文：http://office.microsoft.com/zh-cn/excel-help/HP010090842.aspx?CTT=1
2010	日语：http://office.microsoft.com/ja-jp/excel-help/HP010342659.aspx?CTT=1 中文：http://office.microsoft.com/zh-cn/excel-help/HP010342762.aspx?CTT=1
2013	日语：http://office.microsoft.com/ja-jp/excel-help/HP102749007.aspx?CTT=1 中文：http://office.microsoft.com/zh-cn/excel-help/HA102748996.aspx?CTT=1

数据总览

这次我们来分析虚构的北海道餐馆“腊舞思”的销售数据（除了笔者正好在北海道进行写作以外，这个虚构毫无其他深意）。在分析阶段①设定的目标里，“腊舞思”的经营者考虑到进货以及将来的发展，希望能够预测销售额。考虑到来餐馆的客流量和气温可能有关，我们来分析气温和销售额的关系。

表2的数据是气温和销售额的信息。我们来用表3的函数来分析这些数据。计算结果如表4所示。这些信息称作“描述统计”。这些集中趋势和离中趋势的分析，可以用于把握趋势或者全局。

我们稍微看一下这些结果数据，就能知道这里面包含了各种信息。比如，从标准方差和平均值可以看出数据的分散程度比较大。描述统计也是日常工作和对话中经常被使用的大家熟悉的数据分析信息。

这些描述统计可以用Excel的分析工具库来计算。从“数据”菜单启动“分析工具库”。在打开的用于选择想要使用的功能的对话框里选择“描述统计”。在“输入区域”里选择气温和销售额的列，选择“汇总统计”复选框，按下“确定”按钮（图4）。确认一下是否会显示出表5的结果。尽管数据的格式会有有所差异，如果改变表格的格式的话，应该能得到和表4

▼ 表2　餐馆“腊舞思”的销售数据

月日	气温（℃）	销售额（千日元）
6/1	12	1800
6/2	9	1080
6/3	15	2340
6/4	21	3465
6/5	24	3816
6/6	20	3120
6/7	11	1617
6/8	12	1836
6/9	24	3758
6/10	16	2520
6/11	9	1040
6/12	10	1440
6/13	15	2489
6/14	15	4500
6/15	16	2489
6/16	20	3000
6/17	19	2850
6/18	14	2100
6/19	22	3498
6/20	14	2489
6/21	10	1446
6/22	12	1602
6/23	11	1452
6/24	12	1638
6/25	11	1132
6/26	13	472
6/27	19	2844
6/28	24	3686
6/29	24	3154
6/30	24	3722

▼ 表3　描述统计和函数

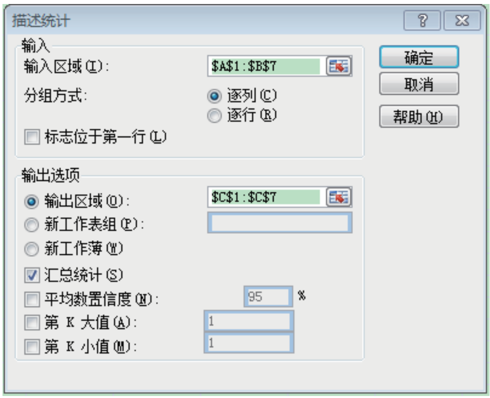
项目	Excel函数	可以得到的结果
最小值	min{范围}	—
最大值	max{范围}	—
合计	sum{范围}	—
平均值	average{范围}	可以得到把所有数加起来的中心值。受极端大或者极端小的数据的影响
中央值	median{范围}	可以得到把数据按大小排序后的中心值。在有极端大或者极端小的数的情况下，这个值比较正确
最频值	mode{范围}	可以得到出现次数最多的值。如果最频值有多个的话，数据里有可能含有不同类型的数据
标准方差	stdev{范围}	可以得到每个值和平均值之间的差。用于了解数据的分散情况。方差小表明数据比较集中，方差大表明数据比较分散

相同的数值。使用分析工具库，离差、峰度、偏度等描述统计也自动被计算了出来。

▼ 表4　描述统计的计算结果

	气温(℃)	销售额(千日元)
min(范围)	9	472
max(范围)	24	4500
sum(范围)	478	72394
average(范围)	16	2413
median(范围)	15	2489
mode(范围)	24	2489
stdev(范围)	5	1019

▼ 图4　描述统计的设置



制作散点图

接着我们制作散点图，来使数据的分散情况视觉化。从菜单的“插入”选项中选择“图标”中的“散点图”（图5）。与单纯的数据比起来，散点图更为直观。在数据分析里面，数据的呈现方式也很重要。数据的视觉化技术应该引起大家的注意。

制作直方图

通过制作直方图可以更清楚地了解数据的核心部分。我们来制作一下销售额数据的直方图。这需要使用我们安装好的分析工具库。和前面一样，从“数据”菜单启动“数据分析”，选择“直方图”。

在输入范围里选择销售额。选择“制作图形”

Note　数据分析的参考网站

除了这次的测试数据以外，读者可以访问“科学工具箱”网站（链接2），获取学习用数据。选择MS Gothic（内容）→MS Gothic（数据资源库），会发现很多用于学习的数据。这个网站本身也收录了很多学习数据分析的内容，相当不错，希望能够对大家有帮助。

▼ 链接2

网站名	URL
科学工具箱	http://rikanet2.jst.go.jp/contents/cp0530/start.html#

复选框。按下“确定”按钮就会生成表6和图6。

接着我们稍微调整一下直方图。从图6我们可以看出，数据区间是自动计算出来的，不够直观。我们可以参考表6的信息，把它换算成易于理解的单位，像表7那样制作按千单位递增的数据区间列表。

接着重新制作直方图。在“输入范围”下面的“数据区间”里选择表7作成的数据。然后对重新画出来的表，按照下面的步骤进行调整，到达表8和图7的状态就完成了。

1. 把标题设成“销售额的分布”。
2. 删除右边和左边的“频率”的标签。
3. 在棒图的地方单击右键，在格式设定里面把间隔设成“0”。
4. 把“下一级”改写成“5500”。
5. 调整横竖的比例，到合适为止。

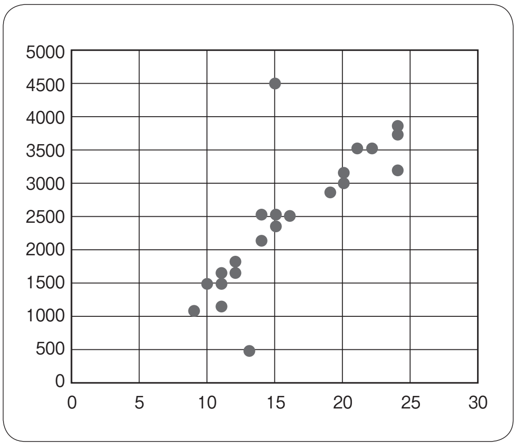
清除异常值

制作了散点图和直方图以后，我们可以发现里面有一些异常值。以上面的数据为例，有时销售额会出现异常高或异常低(图8)。存在这样的数据，尽管是特例，但也会对整体结果产生影响，一般情况下应该删除这些数据。

▼ 表5　用分析工具库生成的描述统计

	气温(℃)		销售额(千日元)
平均值	15.93333333	平均值	2413.136667
标准误差	0.934564049	标准误差	186.1031263
中央值	15	中央值	2488.75
最频值	24	最频值	2489
标准方差	5.118818111	标准方差	1019.328803
分散	26.20229885	分散	1 039 031.209
尖度	−1.246013172	尖度	−0.873000538
偏度	0.396655301	偏度	0.1082428
范围	15	范围	4028.1
最小值	9	最小值	471.9
最大值	24	最大值	4500
合计	478	合计	72 394.1
样本数	30	样本数	30

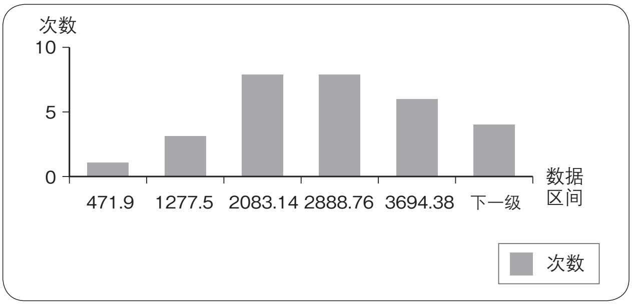
▼ 图5　气温和销售额的散点图



▼ 表6　销售额的直方图(数据区间)

数据区间	频率
471.9	1
1277.52	3
2083.14	8
2888.76	8
3694.38	6
下一级	4

▼ 图6　销售额的直方图(图形)



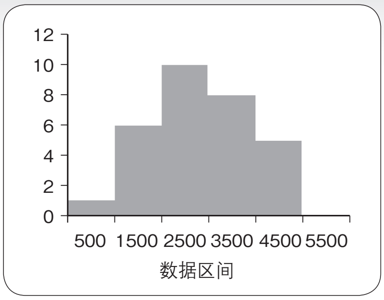
▼ 表7 调整后的数据区间

数据区间
500
1500
2500
3500
4500

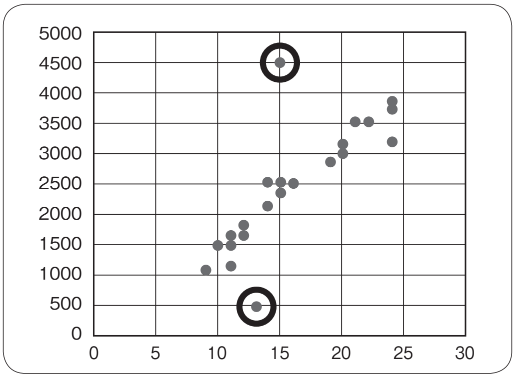
▼ 表8 加工后的销售额直方图（数据区间）

数据区间	频率
500	1
1500	6
2500	10
3500	8
4500	5
5500	0

▼ 图7 加工后的销售额直方图（图表）



▼ 图8 异常值



在这里我们采取的方法是清除异常值，有时会采取修正的方式。比如值缺失（称为“缺损值”），就需要恰当处理。有时假定成平均值比较合适，有时干脆清除掉。根据分析的内容和方法，必须恰当处理异常值。

分析数据（Excel篇）

如前所述，在数据整理的同时，其实也在进行数据分析。下面我们来导出公式。之前我们动手整理好了数据，现在来稍微学一些统计分析知识，这是数据分析的基础。

什么是统计分析

我们已经提到了统计、统计学、统计分析等术语，对于统计这个词大家多少有所了解。统计就是从数量上来把握集合（数据集合）的属性，并用数值和图表来予以呈现。把统计系统化的学问就是统计学，根据统计学进行分析就称为统计分析。

描述统计和推论统计

在调查对象都已全部准备好的情形下，对这些全部的数据即母集合（population）进行的分析属于描述统计学（descriptive statistics）的范畴。在只有一部分调查对象准备好的情形下，对这一部分数据即样本（sample）进行分析，进而推测母集合，属于推论统计学（inferential statistics）的范畴。在现实中，能把握全部调查对象的情形很少，所以应用的多是推论统计。数理统计学以及多变量分析能区别其不同的关联性，在此我们不作赘述。

定量数据和定性数据

数据可以分为表示数量的定量数据和表示分类的定性数据。可以说气温或者体重是定量数据，天气和性别则是定性数据。把这些数据再进行细分的例子，如表9所示。

这些分类对于选择分析方法非常重要。统计学入门的网站（链接3）上公开了用于不同数据类型的分析方法，想全面掌握的读者不妨看一下。

相关

相关就是从现象上推论两种数据是否按照相同的趋势变动。表示相同趋势变动程度的值叫做“相关系数”。相同趋势只表明相互的统计关联，并不表示一方是原因另一方是结果的因果关系。比如，从相关系数上我们知道“气温”和“销售额”有相同的变动趋势，但原因是什么就必须人为分析。

如表10所示，根据两种数据的分类，不同类型统计关联的分析计算方法是不同的，这一点需要注意。由于这次分析的是定量数据和定量数据，所以我们采用单相关系数。

这个值用Excel函数可以很容易算出。对清除了异常值的气温和销售额数据应用CORREL函数。指定第一变量为气温的值，指定第二变量为销售额的值，结果是0.9767782，表明两者有非常强的关联关系。这里必须注意，前文已经说过的，有关联关系并不意味着有因果关系。在这次的例子里，尽管数值显示随着气温上升销售额也会上升，但并没有显示气温上升是销售额上升的原因，我们还需要考虑其他因素。

回归分析

对气温和销售额进行关联分析，计算出相关系数后，尽管可以知道两者的关系，但还不能进行预测。接着，再进行用于预测分析的回归分析。在气温（x）是原因，销售额（y）是结果的情形下，把x对y的影响用线性关系表示出来。这时x叫做说明变量（explanatory variable）或者自变量，y叫做目的变量（criterion variable）或者因变量。这次仅仅是用Excel来导出回归公式，实际的回归分析是按照下面的流程实施的。

▼ 链接3　统计学入门网站

网站名	URL
统计学入门	http://www.snap-tck.com/room04/c01/stat/stat02/stat0201.html

▼ 表9　数据分类

	分类	说明	例子
定量数据	比例尺度（ratio scale）	数值之间存在比例关系，可以进行四则运算（加减乘除）	金额，速度，时间
	间隔尺度（interval scale）	存在顺序关系，连续的，可以进行加减运算	温度，公历
定性数据	顺序尺度（ordinal scale）	存在大小的顺序关系	满足度，感情，态度
	名义尺度（nominal scale）	保持着合理性	血型，性别，出生地

▼ 表10　数据类型组合对应的分析方法

组合	指标	可取得的值
定量数据和定量数据	单相关系数	−1～1
定量数据和定性数据	相关比	0～1
定性数据和定性数据	克拉默相关系数	0～1

1. 绘制目的变量和说明变量的散点图。
2. 导出回归公式。
3. 确认回归公式的精度。
4. 推测母集合。
5. 进行预测。

进行单回归分析

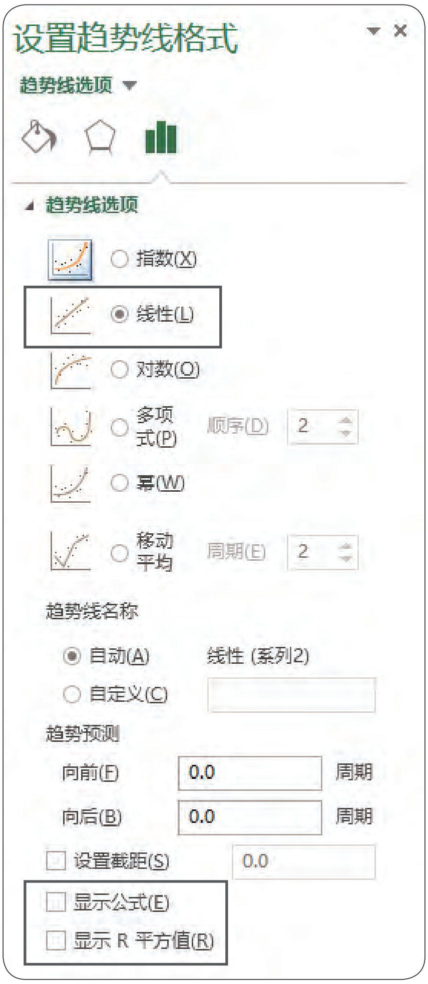
只有一种说明变量的回归分析称作单回归分析，我们用Excel来实施这个分析。

➡ 添加趋势线

用清除了异常值得数据重新制作散点图。在这个散点图上画趋势线。趋势线是由回归分析导出的回归公式（y=ax+b）的图形。选择图形，按照链接 4 的步骤添加趋势线。选项上选择“线性”，在复选框里选择“显示公式（E）”和“显示R 平方值（R）”。

Excel 2013的界面发生了很大变化，具体步骤如下。选择图表，单击图表右上角的“图表元素”按钮“+”，选中“趋势线”框。单击“趋势线”旁边的箭头，单击“其他选项”。在右边会显示出设定，单击图形的图标，在“趋势线格式”窗格中的“趋势线选项”下选择“线性”，在复选框里选择“显示公式（E）”和“显示R 平方值（R）”（图9）。

▼ 图9　趋势线的选项



▼ 链接4　添加趋势线

版本	URL
2003	日语：http://office.microsoft.com/ja–jp/excel–help/HP005198462.aspx
	中文：http://office.microsoft.com/zh–cn/excel–help/HP005198462.aspx?CTT=1
2007	日语：http://office.microsoft.com/ja–jp/excel–help/HP010007461.aspx?CTT=1#BMaddtrendline
	中文：http://office.microsoft.com/zh–cn/excel–help/HP010007461.aspx?CTT=1
2010	日语：http://office.microsoft.com/ja–jp/excel–help/HP010342158.aspx
	中文：http://office.microsoft.com/zh–cn/excel–help/HP010342158.aspx?CTT=1
2013	日语：http://office.microsoft.com/ja–jp/excel–help/HP102809798.aspx
	中文：http://office.microsoft.com/zh–cn/excel–help/HA102809798.aspx?CTT=1

图10里的公式就是回归公式。由气温（x）的值能够计算出销售额。把预想的气温值乘以167.35再减去281.55就能得到销售额的预测值。R平方值，简单说就是表示回归公式精度的值。越接近1精度越高，例子中的值表示得到的回归公式的可信度非常高。

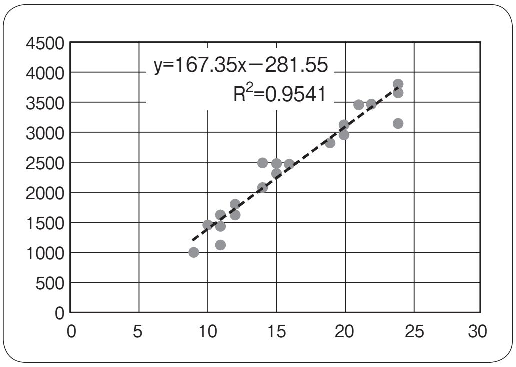
也可以用和趋势线不同的方法计算回归公式。从菜单启动分析工具库，选择“回归分析”，在“因变量y的范围”里选择销售额的值，在“自变量x的范围”里选择气温的值。按下“确定”按钮，就会显示出分析结果。可以看出，图11里背景色不同地方的值和前面图表里的值是一致的。

另外，在实际分析工作中，有时应使用修正R平方值，有时使用t或者P值，版面所限，在此不作详述。

小结

本章我们讲解了数据分析的基础知识及用Excel进行数据分析的一些方法。在接下来的章节里，我们会介绍使用更加专业的数据分析工具的分析方法。

▼ 图10　趋势线和回归公式



▼ 图11　使用分析工具库得到的回归分析结果

回归统计

Multiple R	0.976778243
R Square	0.954095735
Adjusted R Square	0.952330187
标准误差	197.1300769
观测值	28

方差分析

	df	SS	MS	F	Significance F
回归分析	1	20999939.86	20999939.86	540.3961773	6.36544E-19
残　差	26	1010366.948	38860.26722		
总　计	27	22010306.8			

	Coefficients	标准误差	t Stat	P-value	Lower 95%	Upper 95%	下限 95.0%	上限 95.0%
Intercept	-281.5523909	121.545944	-2.316427695	0.028679251	-531.393657	-31.71112486	-531.393657	-31.71112486
X Variable 1	167.3477043	7.198858297	23.24642289	6.37E-19	152.5502392	182.1451695	152.5502392	182.1451695

CHAPTER 2 ②

数据分析入门必备知识

让数据分析工具助你一臂之力 [R · Mahout 篇]

承接前一章，我们来进行数据分析的实践。本章使用更加专业的分析工具R和Mahout。

文/高木基成　三得利系统技术株式会社　译/杨驹武

分析数据（R篇）

上一章，我们用Excel进行了单回归分析。有两种以上解释变量的回归分析叫做多元回归分析。尽管也可以用Excel做多元回归分析，但这次我们用R来分析。

R是什么

R（也叫R语言）是用于统计分析的开源软件。包括统计分析的编程语言及其运行环境。在教学和实际工作中普及很广。在提供的扩展包里包含了最新的研究成果和方便的功能。R本体是用CUI界面（含有简单的GUI）操作的。复杂的分析需要编写程序。R的高级CUI环境有RCommander和RStudio。这里我们只使用R单体。

安装R

首先下载R的模块。官方网站里有镜像服务器的链接。比如可以从兵库教育大学的网站下载（链接1）。

安装方法和通常软件一样，在Windows或者Mac操作系统下，只要点击几下鼠标就可完成安装，不会有什么麻烦（链接2）。

尝试R

接下来，启动R试用一下。从R的图标启动就会打开R的运行窗口（图1，画面是Windows

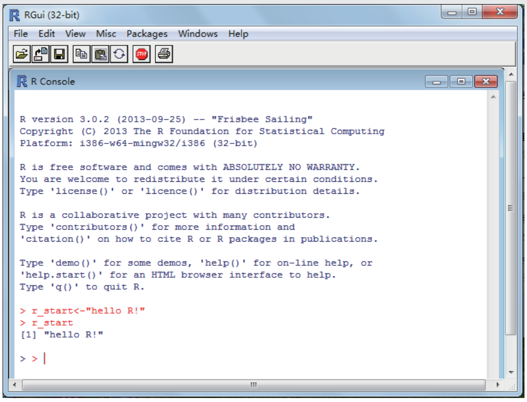
▼ 链接1　下载R模块的网站

服务器	URL
兵库教育大学	http://essrc.hyogo-u.ac.jp/cran/

▼ 链接2　安装方法

OS	URL
Windows	http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-Windows
Mac	http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-OS-X
Unix	http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-Unix_002dalikes

▼ 图 1 R的运行界面



版的)。R不用定义类型。输入r_start<-"hello R!" 按下回车键，输入r_start，按下回车键后就会显示hello R!。

多元回归分析

表 1 是在我们分析过的“腊舞思”的销售数据里，加入餐馆前通过的行人数(清除了前面章节分析出的异常值)。我们就用这些数据来分析。

在 R 的运行窗口里输入图 2 的内容。确认运行这些命令是否能得到如图 3 和图 4 的结果。图 3 是散点图，可以从视觉上了解各个变量的关联关系。向右上倾斜表示正相关，向右下倾斜表示负相关。图 4 的 Intercept 称作切片，表示和坐标轴的交点。air 和 passer 表示对气温和行人变量的倾斜，是偏回归系数。用这些值来构建多元回归公式。新的多元回归公式如下所示。

$$y(销售额) = x1(气温) \times 162.1573 + x2(行人) \times 0.1141 - 290.0759$$

在实际分析工作中，经常要探讨几十种变量。变量多了，要考虑的内容就会增加。所以构建解释变量尽量少，且精度高的多元回归公式非常重要。构建多元回归公式的方法有下面几种。

- 变量增加法。
- 变量减少法。
- 变量增减法。
- 基于信息量准则的方法。

使用 R 的 step 函数，可以基于信息量准则，对变量进行研究。运行图 5 的命令。从结果可

▼ 表 1 餐馆“腊舞思”的销售数据(+行人)

月日	气温(℃)	行人	销售额(千日元)
6/1	12	625	1800
6/2	9	350	1080
6/3	15	575	2340
6/4	21	1195	3465
6/5	24	1185	3816
6/6	20	825	3120
6/7	11	575	1617
6/8	12	285	1836
6/9	24	1425	3758
6/10	16	825	2520
6/11	9	577	1040
6/12	10	565	1440
6/13	15	805	2489
6/15	16	825	2489
6/16	20	975	3000
6/17	19	975	2850
6/18	14	685	2100
6/19	22	1325	3498
6/20	14	825	2489
6/21	10	425	1446
6/22	12	605	1602
6/23	11	585	1452
6/24	12	645	1638
6/25	11	975	1132
6/27	19	875	2844
6/28	24	1025	3686
6/29	24	875	3154
6/30	24	1125	3722

▼ 图2　R命令

```
❶ 读入气温值
air<-c(12,9,15,21,24,20,11,12,24,16,9,10,15,16,20,19,14,22,14,10,12,11,12,11,19,24,24,24)

❶ 读入行人数
passer<-c(625,350,575,1195,1185,825,575,285,1425,825,577,565,805,825,975,975,685,1325,825,425,605,585,
645,975,875,1025,875,1125)

❶ 读入销售额
sales<-c(1800,1080,2340,3465,3816,3120,1617,1836,3758,2520,1040,1440,2489,2489,3000,2850,2100,3498,2489,
1446,1602,1452,1638,1132,2844,3686,3154,3722)

❶ 建立数据集
rausudata<-data.frame(air,passer,sales)

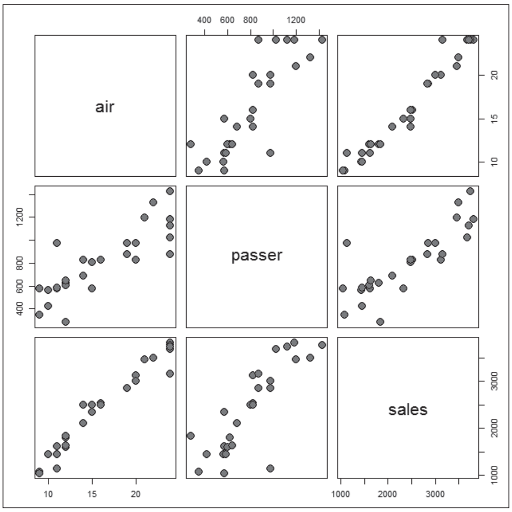
❶ 表示数据集的内容
rausudata

❶ 建立相关系数的行列式
round(cor(rausudata),4)

❶ 用散点图确认全体的关系
pairs(rausudata,pch=21,bg="red",cex=2)

❶ 用销售额以外的信息进行多元回归分析
(rausudata.lm<-lm(sales~.,data=rausudata))
```

▼ 图3　散点图



▼ 图4　R命令的运行结果

```
Call:
lm(formula = sales ~ ., data = rausudata)

Coefficients:
(Intercept)         air            passer
   -290.0759     162.1573          0.1141
```

以看出，对于本次数据，最合适的模型是只使用气温的单回归公式。

相信大家已经明白，用R语言可以很方便地计算多元回归分析。在实际工作中数据分析不会这么简单，要考虑会的变量很多，必须探讨诸如应该采用哪个变量，回归公式的精度有没有问题等。

▼ 图5　step函数及运行结果

```
> rausudata.lm2<-step(rausudata.lm)

Start:  AIC=299.57
sales ~ air + passer

         Df Sum of Sq     RSS    AIC
- passer  1      8552 1009954 297.81
<none>                1001403 299.57
- air     1   5889519 6890922 351.58

Step:  AIC=297.81
sales ~ air

        Df Sum of Sq      RSS    AIC
<none>                1009954 297.81
- air    1  20995991 22005945 382.09
```

处理大数据

到现在为止，我们还没有分析过“大数据”类型的大量数据。不过不涉及大数据也有很多可以分析的事情。在实际的分析工作中，要根据设定的目标，慎重判断要分析数据到哪一层次。但是，就像分类垃圾邮件那样，没有大量的数据分析就不能得到期望的结果。这种情况下，就要用到大型数据处理平台Mahout了。

机器学习

机器学习(Machine Learning)是人工智能(Artificial Intelligence)的一个研究领域，目标是让计算机拥有和人一样的学习能力。比如“从多个图像里面只抽取出有苹果的图像”，机器学习大致会按照下面的步骤来实现。

1. 准备用于识别苹果的图像样品。
2. 从图像样品学习特征(颜色以及形状等)，开发抽取和管理的程序。
3. 使用学到的信息，对大量照有苹果的图像进行判断。

不是开发直接解决问题的处理程序，而是开发“学习怎么解决问题的方法”的处理程序。这里的学习以及判断都要用到统计，所以也叫做统计式机器学习，可以说统计和机器学习的界线已经很模糊了。

与一般统计解析相比，适合用机器学习的领域有如下特征：①学习用的数据越多准确性越高(能得到期望的结果)。②随着要处理的数据的增多，计算量也增加(处理时间长)。为了加强机器学习的能力，兴起了用处理大数据方法进行机器学习的潮流。

两种学习

机器学习的问题大致可分为“监督学习”(Sapervised learning)和“无监督学习”(unspervised learning)两类。尽管也有介于两者之间的“部分监督”以及“强化学习”等种类，但前两类是最基本的，必须加以理解。这里我们挑战“监督学习”这一类。

➡ 监督学习

“监督学习”是指用多个叫做学习数据(也叫做训练数据)的配套输入输出的样品数据进行学习。运用学习的结果，一旦输入数据就能正确地输出预测结果。因为这种处理所需要的正确数据，来源于知道正确答案的教师那里，所以叫做“监督学习”。

➡ 无监督学习

“无监督学习”是指只提供输入数据。和监督学习不同，此种情况下，没有给出用于判断的正确数据，适用的领域自然也受限制。没给正确解答，就必须设定某种判定标准来进行处理。也许有人会认为“监督学习”比较好用，但准备正确答案数据的费用很高，近几年针对“无监督学习”的研究很活跃。

Mahout

有许多读者已经对Mahout有所了解，在此再简单地介绍一下。Mahout是用于机器学习的程序库，是由Apache软件基金会开发的开源软件(链接3)。

图6是Mahout的架构，算法大致分为5类，其使用实例有，商品推荐(Recommenders)、新闻组聚类(Clustering)、邮件软件的垃圾邮件分类(Classification)等。高频商品组合数据挖掘法(Frequent pattern mining)方面的实例如，通过POS终端数据得出“同时购买啤酒和婴儿尿布的顾客比较多”的信息。商品送货车配车计划的最优化，则是遗传算法(Genetic)的应用实例。

Mahout的机器学习算法超过40个，因其可用于分散处理海量数据的平台Apache Hadoop（链接3），近几年受到极大关注。

构建环境

接着，我们按照以下步骤，来构建Mahout的运行环境。如果是Mac或者Linux操作系统，很快就可以构建环境，如果是Windows操作系统，要先安装Cygwin，以便shell可以运行。由于Windows操作系统麻烦一些，笔者在Amazon EC2上用CentOS操作系统构建环境。

1. 安装Java 6
   - 添加Java到运行路径里，运行Java-version，使之能显示Java的信息。
   - 也可以用yum search jdk搜索OpenJDK，然后安装。
2. 安装Apache Maven
   - 添加Maven到运行路径，运行mvn-version，使之能显示Maven的信息。
3. 安装Apache Mahout

我们从第三步的Mahout的安装开始解说。笔者撰写本稿时，Mahout的最新版本是0.7。

从链接4下载官方版本mahout-distribution-0.7-src.tar.bz2。下载好以后按照图7的步骤安装。

用Mahout进行分类的理由

正如前文所述，Mahout的特点是可用于Hadoop 。分类属于有教师的机器学习，因而

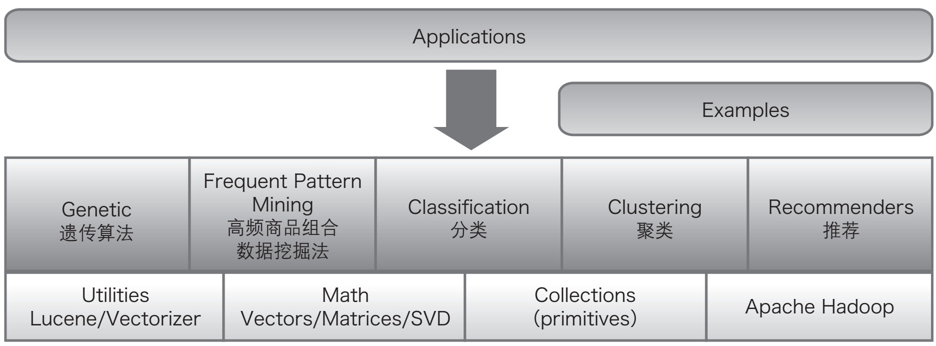
▼ 链接3　下载R模块的网站

网站名	URL
Java SE Downloads	http://www.oracle.com/technetwork/java/javase/downloads/index.html
Apache Maven	http://maven.apache.org/
Apache Mahout	http://mahout.apache.org/
Apache Hadoop	http://hadoop.apache.org/

▼ 链接4　Mahout下载网站

网站名	URL
Apache Mahout	https://cwiki.apache.org/confluence/display/MAHOUT/Downloads

▼ 图6　Mahout的架构



※参考：http://www.slideshare.net/hamadakoichi/mahout-recommendation-mahout-jp-tokyowebmining-14th
https://cwiki.apache.org/confluence/display/MAHOUT/Algorithms

需要准备大量的学习数据。在学习数据超过100万项的情况下，单单处理学习就要花很多时间。但如果在Hadoop上运行，时间就可以缩短。而且Hadoop能有效缩短实际运行时处理分类的时间。

当然，如同这次的示例，没有Hadoop，Mahout也能独立运行。Mahout是用Java开发的，在维护以及API的易用性上也有优势。

Mahout的分类算法

Mahout包含了以下分类算法。这里使用朴素贝叶斯算法。Mahout实现的朴素贝叶斯算法只适用文本格式的数据。大致来讲，可以有效地处理100万项以上的数据。

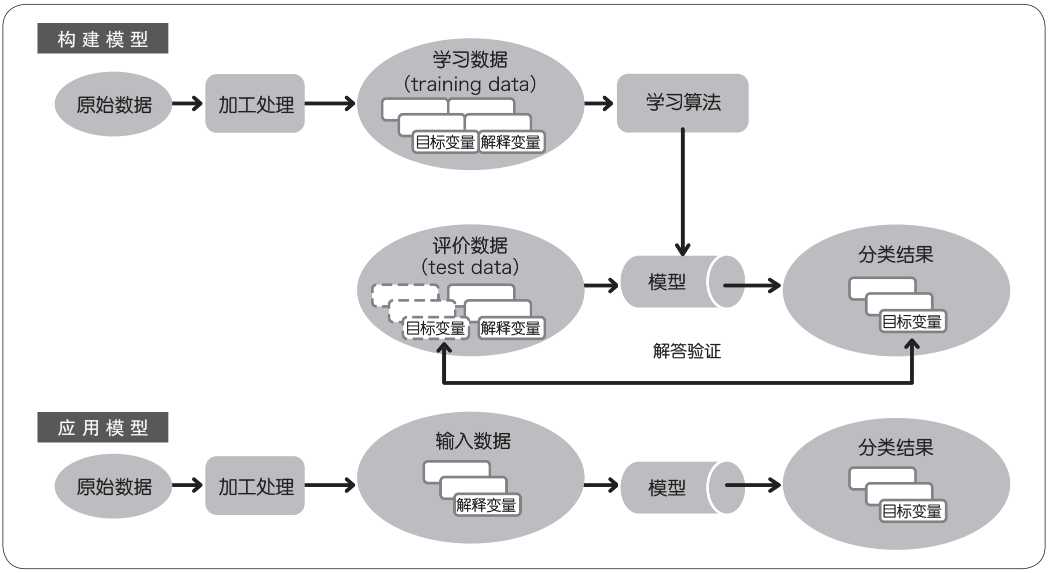
1. 随机梯度下降（Stochastic Gradient Descent）OnlineLogistic, Regression, AdaptiveLogistic Regression, CrossFoldLearner
2. 支持向量机（Support Vector Machine）
3. 朴素贝叶斯（Naive Bayers）
4. 增强型朴素贝叶斯（Complementary naive Bayers）
5. 随机森林（Random forests）

*上面的五种方法，数字越大，学习成本越高。

▼ 图7　安装Mahout

❶ 解开压缩的模块
```
$ tar jxvf mahout-distribution-0.7-src.tar.bz2
```
❷ 移动到解开压缩后的文件目录
```
$ cd mahout-distribution-0.7
```
❸ 用Maven命令安装特别花时间。用mvn install –Dmaven.test.skip=true这样省略测试的命令安装的话会快一些（但不推荐使用）
```
$ mvn install
```
❹ 运行mahout命令
```
$ bin/mahout
```

▼ 图8　分类作业的流程



流程

接下来按照图8所示来实现分类。首先准备用于学习以及用于评价学习结果的数据。把这些数据加工成目标变量（想要判断的分类结果）和解释变量，分为用于学习的数据和用于评价的数据。用学习数据构建模型，用评价数据评价模型的正确性。评价数据里包含正确的目标变量，可以用来验证分类结果推测的目标变量。

分类

这次的学习数据我们用Wikipedia。日语数据需要用MeCab进行形态分析加工处理。简单起见，我们用英语的内容。

使用表2的URL，把浏览器里显示的文本全部选中、复制，然后保存到文件里做成测试数据。完成后复制到Mahout的安装目录下。比如Mahout的安装目录是/home/mahout/mahout-distribution-0.7，那么就把数据复制到

▼ 表2　用于学习的内容一览表

目录		文件（.txt）	URL	内容
wiki	math	mathematics	http://en.wikipedia.org/wiki/Mathematics	数学
		probability	http://en.wikipedia.org/wiki/Probalility	概率论
		statistics	http://en.wikipedia.org/wiki/Statistics	统计学
		vector	http://en.wikipedia.org/wiki/Vector_calculus	向量分析
		algebra	http://en.wikipedia.org/wiki/Algebra	代数学
		geometry	http://en.wikipedia.org/wiki/Geometry	几何学
		analysis	http://en.wikipedia.org/wiki/Mathematics_analysis	解析分析
		set	http://en.wikipedia.org/wiki/Set_thoery	集合论
		info	http://en.wikipedia.org/wiki/Information_engineering	信息工程
		calculus	http://en.wikipedia.org/wiki/Calculus	微积分
	jura	jurisprudence	http://en.wikipedia.org/wiki/Jurisprudence	法学
		judiciary	http://en.wikipedia.org/wiki/Judiciary	司法
		legislation	http://en.wikipedia.org/wiki/Legislation	立法
		public	http://en.wikipedia.org/wiki/Public_law	公法
		civil	http://en.wikipedia.org/wiki/Civil_Law_（area）	民法
		criminal	http://en.wikipedia.org/wiki/Criminal_law	刑法
		international	http://en.wikipedia.org/wiki/International_law	国际法
		philosophy	http://en.wikipedia.org/wiki/Philosophy_of_law	法律哲学
		history	http://en.wikipedia.org/wiki/Legal_history	法学史
		comparative	http://en.wikipedia.org/wiki/Comparative_law	比较法学
	medi	medicine	http://en.wikipedia.org/wiki/Medicine	医学
		wmedicine	http://en.wikipedia.org/wiki/Western_medicine	西医学
		tcmedicine	http://en.wikipedia.org/wiki/Traditiona_Chinese_medicine	中医学
		cardiology	http://en.wikipedia.org/wiki/Cardiology	循环器学
		gastroenterology	http://en.wikipedia.org/wiki/Gastroenterology	消化器学
		pulmonology	http://en.wikipedia.org/wiki/Pulmonology	呼吸器学
		nephrology	http://en.wikipedia.org/wiki/Nephrology	肾脏学
		endocrinology	http://en.wikipedia.org/wiki/Endocrinology	内分泌学
		hematology	http://en.wikipedia.org/wiki/Hematology	血液学
		neurlogy	http://en.wikipedia.org/wiki/Neurology	神经学

▼ 图9　分类处理的运行步骤

❶ 把测试数据转换成顺序文件
（生成Key是目录、Value是单词的数组）
```
bin/mahout seqdirectory -i wiki -o wiki-seq
bin/mahout seq2sparse -i wiki-seq -o wiki-vectors -lnorm -nv -wt tfidf
```
❷ 分开用于学习的数据和用于评价的数据
（指定randomSelectionPct为20%来分割测试数据）
```
bin/mahout split -i wiki-vectors/tfidf-vectors --trainingOutput wiki-train-vectors --testOutput
wiki-test-vectors --randomSelectionPct 20 --overwrite --sequenceFiles -xm sequential
```
❸ 学习处理
```
bin/mahout trainnb -i wiki-train-vectors -el -o wiki-model -li labelindex -ow
```
❹ 评价学习结果
```
bin/mahout testnb -i wiki-test-vectors -m wiki-model -l labelindex -ow -o wiki-testing/
```

/home/mahout/mahout-distribution-0.7/wiki下。

运行图9的命令进行分类处理。图9命令4运行后，会显示出如图10的分类结果。在这个结果里，评价了大约50%的18个文件。正确分类的有16个，准确率非常高，约90%。两个医学文件被错分到了数学，但没有医学、数学、法律大分类上的错误。

在实际开发时，要解决的问题会复杂得多，要达到这样的准确性要费很多工夫。这次用朴素贝叶斯算法对Wikipedia的内容进行了少许分析，请大家一定用别的算法和数据试试，或者在Hadoop上试试，从不同的角度学习Mahout。

▼ 图10　分类结果

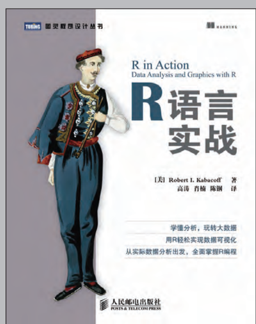
```
Summary
-------------------------------------------------------
Correctly Classified Instances   : 16   88.8889%
Incorrectly Classified Instances : 2    11.1111%
Total Classified Instances       : 18

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       <--Classified as
4       0       0        |  4       a       = jura
0       7       0        |  7       b       = math
0       2       5        |  7       c       = medi
```

小结

这一章我们动手用R进行了多元回归分析，也用Mahout处理了大数据。大家应该对数据分析软件的操作有一定了解了吧。请大家以这次的学习为契机，尝试用Excel、R和Mahout进行各种各样的数据分析。

延伸阅读



ISBN：978-7-115-29990-1
定 价：79.00元
页 数：408页

CHAPTER 3

不用数学公式也能学？

如何掌握机器学习

本章作为Cybozu · Live株式会社员工的作者，将通过公司内部学习，相关学习书籍，进一步介绍成为数据科学专家的预备知识及学习方法。

文/竹迫良范　Cybozu Live株式会社　takesako@shibuya.pm.org
译/雷军

数据科学专家

大数据形势下，我们需要从企业内部积蓄已久的庞大数据中，抽取出对经营有益的信息，并客观地分析这些信息，做出必要的决策。因此，与企业的效益密切相关的“数据科学专家”越来越受瞩目。

美国《哈佛商业评论》[①]甚至预言：“数据科学专家是一颗巨星，是21世纪最有魅力的职业。”在日本，只有部分企业有专门的数据科学专家，即使在这种条件下，为什么这个职业还是备受关注呢？

用户数量和使用时间趋于饱和

数据分析类职业受到瞩目的理由之一是“国内市场已经饱和”。日本的人口约为1.2亿，据2011年统计，网民的人口比例为79%，约9610万人[②]。

拥有数千万用户的互联网服务，其可以预见的成长前景是怎样的呢？且不说国内人口能否增长，注册账户数量的增长也会趋于饱和。为了打开局面，企业需要向国际市场推进，销售战略不能仅限于日本市场，或者，实施一些销售策略，使用户使用多个账户，以促进用户数量增长。比如，手机的普及率超过人口比率的100%，一个人同时持有普通手机和智能手机，或者私用手机和公务用手机，诸如此类。同样地，电子邮箱也有一人拥有多个的情况，这是很普遍的现象。

以前的游戏机，连接电视机后才能玩游戏，所以只有在家时才能玩。但是，近几年移动终端上的社交游戏，不论在家中、上班上学途中、在学校、在公司，只要是休息时间都可以玩。这种模式扩大了游戏时间及地点，从而带动用户量增加，成功地扩大了市场。

但是最近，随着日本企业向国外推进，人们也逐渐意识到一个事实，那就是世界的人口总数（大约是71亿）并不是无限的，每个人能每天的时间最多也只有24小时。无论是用户数量还是使用时间最终总会到达饱和状态，为了提高企业的效益，增加每个用户的消费就成为必须要考虑的策略。

① Thomas H. Davenport and D. J. Patil, “Data Scientist: The Sexiest Job of the 21st Century”, Havard Business Review, Oct., 2012.
② 出自总务省“平成23年通信利用动向调查”http://www.soumu.go.jp/johotsusintokei/statistics/statistics05.html。

向上销售

这是一个销售战略的术语。这种营销策略是根据客户的消费喜好，提供更高价值的产品或服务，刺激客户更多的消费，增加每个客户产生的效益。例如，一名客户购买电脑时，如果能促使客户购买配备了最新英特尔CPU的电脑，那就是向上销售。

交叉销售

这也是一个销售战略的术语，目的是劝诱已有的客户购买其他的商品，以促使单个客户的销售额增加。例如，在快餐店里，当客户点汉堡时，服务员会询问是不是也来份薯条，这样的劝说就是一种交叉销售。还有，在亚马逊网上购物，当客户把想订购的商品加入购物车时，页面上就会出现“购买此商品的客户还订购了这个商品”的推荐。这些都是典型的交叉销售。

为了显示出劝诱的商品，至关重要的是分析过去的销售数据，并从数据中发现密切相关的商品之间的搭配。

到目前为止，商店的店员采用的交叉销售方式，主要是根据各种商品的销售情况，凭直觉摆放商品，以此提高销售额。但是，现在可以利用收银机的记录分析POS数据，有些无法依靠店员的经验和直觉发现的数据间的隐秘关系，就可以被发现了。就像很有名的那个传说，根据POS数据的分析“中年男性购买纸尿布的同时，买罐装啤酒的可能性很高”。

但是，在分析商品之间的关联性时，必然就需要分析全部商品排列组合的频率。如果有一万个商品，每两个商品排列组合起来就会出现五千万种组合，如果商品增加10倍，那么排列组合就会增加100倍。

数据量越庞大，就越有可能发现人们直觉所忽视的意外的数据间的关系，但是，数据量的规模也膨胀到普通计算方法下，一台计算机不可能处理的程度。为了能够在有限的时间内完成计算，必须采用Hadoop或Mahout进行分布式处理，问题才能得到解决。

KPI指标

在销售战略中有一个评价业绩的重要指标叫KPI（Key Performance Indicator，关键绩效指标），它主要取决于访问次数、有效用户数以及ARPU（参加者每人每月销售额）等，可以定期评价是否达到了目标数值。在分析数据时，首先需要判断KPI指标是否达成。

在这里还有个比KPI更重要的概念，即KGI（Key Goal Indicator），它是关键目标指标，重要程度位于KPI之上，是决定最终目标的指标。为了达到KGI的目标，必须具体设定可实现的各阶段的中间目标KPI，短期目标KPI达成下，若无法实现长期稳定的效益增长，企业的经营也仍然是不健全的。

对于数据科学专家来说，需要牢记KPI之上还有KGI，以此积极参与企业经营，这一点非常重要。

学习数据分析的基础：统计

如果上司突然说，“从明天开始你去负责数据科学专家的工作”，你会怎么样呢？突然

要做大数据分析恐怕难度很高，你应该时刻准备着，平常就养成认真仔细分析手头小数据的素养。

为此，最低限度的必备知识就是高中学习过的概率统计。由于入学年限不同，这些内容的叫法各异，数学III、数学C、数学B等说法都有，有的人学得很深入能去参加数学能力的“验收考试”，有的人因为其不属于高中义务教育范围，就没有多少了解。总之，不是所有的人都具备这些知识。

没有数学公式的统计学习书

对于数学公式头痛的人，我给你们首先推荐一本书，不用数学公式学会统计的基础知识。

《通过漫画了解护士的统计学：从观察数据到做成具有说服力的资料》

田久浩志
小岛隆矢（著）
小山惠子（图）
BeCom（制作）
欧姆社出版
2006年5月
2310日元（含税）
ISBN=
978-4274066498



虽然有的人看到这个书名就会敬而远之，但是对于那些尚未了解“卡方检验”的人来说，还是先从这本书开始读起吧。

做数据分析之前，必定先要收集数据，该书从制定调查问卷的项目开始，仔细讲解整个过程，不愧为一本好书。详细讲解了名目尺度（男女区别等）、顺序尺度（好坏等）、连续尺度（身高、体重等）的不同，还解说了统计方法需要注意的要点。正如书名所述，它是面向护士讲解统计学的书，所以没有很难的数学公式，只要能用计算器进行简单的加减乘除和平方根运算，就不难理解该书的内容。

➡ 数据分析之前的准备

该书最大的优点是，作者一开始就仔细讲解了学据分析之前的准备工作。

1. 问题是什么？想要做什么？（抽出评价项目）
2. 为什么这样做才好？（大胆质疑）
3. 怎样才能客观地证明？（小心求证）

这些都是分析大数据时常用的思考方法。

在临床上出现了一个领域，就是积极利用统计学的方法，通过分析过去的临床数据，找出最适合的治疗方法。该书的主要学习内容是：无偏见的数据收集方法，从收集数据到图表制作，分布图的制作法，相关的寻找方法以及检验手法。

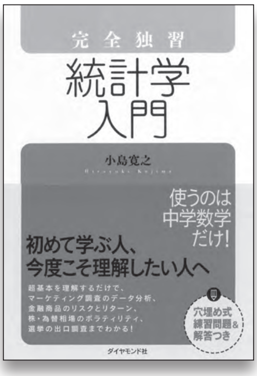
顺便提一下，南丁格尔就是一位用实际业绩和数据证明了“为降低士兵的死亡率必须改善营房卫生环境”的统计学者，她制作了配有多幅图表的报告书，直接成功地领导了医院行政改革。

不使用概率的统计学习书

帝京大学经济学系的小岛宽之先生写的这本书，不用概率而可以学习统计，是一本罕见的入门书。

《完全自学统计学入门》

小岛宽之(著)
钻石出版社
2006年9月
1890日元(含税)
A5 规格
208页
ISBN = 978-4-478-82009-4



统计学大体分成经典的描述统计和推断统计两大部分。

● **描述统计**

掌握总体的特征。如人口调查，土地调查，国情调查。

● **推断统计**

从总体中任意抽取样本，分析抽样，然后推测出全体的情况。如竞选者的当选概率等。

推断统计是现在数据分析使用的基础统计学，是由概率和排列组合结合发展而成的。统计是“记述过去发生的事情”，因为是现实存在的所以容易理解。概率是“描述即将发生的未来的事情”。刚开始学习统计学时，虽然不容易理解二者的区别，但是之后一直都是把统计和概率作为一个整体来学习的。

这本书划时代的一点就是，尽量尝试不用概率说明推断统计。像概率变量$P(X=x1)$、微积分、∑的符号都没有出现。对数学没有信心的人，可以用此书复习统计学。

此书用了很多篇幅说明标准差，可以让学习者更好地体会在实际中如何应用统计进行数字处理。

大数定律与中心极限定理

统计学知识还应用于金融工程领域，学习证券投资的组合构成以及期权价格时，就必须有大数定律与中心极限定律的概念。

● **大数定律**

抽样的数量充分大时，抽样的平均值就可以看作是这个总体的平均值。

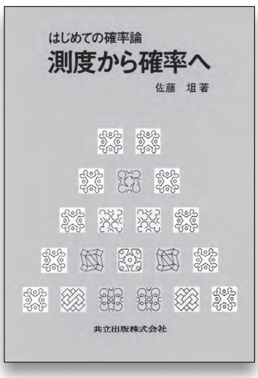
● **中心极限定理**

与总体的分布无关，抽样的数量足够大的话，抽样的平均概率分布就可以作为正态分布。

想要严格地证明大数定律与中心极限定理，那么就有必要学习讲解从柯尔莫哥洛夫开始的近代概率论的教科书。

《概率论初步学习：从测度到概率》

佐藤坦(著)
共立出版社
1994年2月
3360日元(含税)
A5规格，216页
ISBN= 978-4-320-01473-2



该书是面向具有大学一年级数学水平，了解黎曼积分和勒贝格积分的学习者。虽然这本书开始的7页，没有数学基础的人也能读懂，但是之后的内容，若没有学习过集合论基础和实数的连续性就很难懂了。书中的例子讲解，可以让读者有基本理解，但是对于希望更深入理解大数定律和中心极限定理的学习者来说，最好还是自己动手推算一下。

机器学习的模型是以正态分布为假设前提讨论数据的概率分布的，所以理解中心极限定理是必须的。

利用贝叶斯推断预测未来

垃圾邮件过滤就是贝叶斯方法的一个使用例子。通过学习过去观测到的垃圾邮件的词语特征制作分类器，再利用该分类器对新收到的信件进行概率分析和计算推断，判断是否是垃圾邮件。

➡ 贝叶斯定理

大约250年前，一位业余数学家，名叫托马斯·贝叶斯的牧师，发现了“贝叶斯定理”。

$$P(B|A)=\frac{P(A|B)\cdot P(B)}{P(A)}$$

常量：$P(A)$= 现象A发生的概率

先验概率：$P(B)$= 现象A发生前现象B的概率分布

后验概率：$P(B|A)$= 现象A发生后现象B的概率分布

似然函数：$P(A|B)$= 利用观察结果合理地推断前提条件是什么

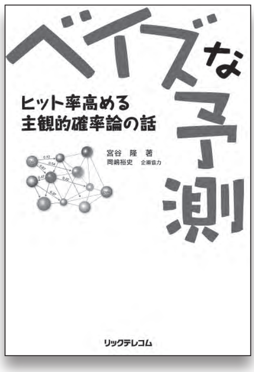
后验概率是指以观察一种现象为动机，高精度修正后的先验概率。

最大似然估计就是选择假设最大化似然函数的推断方法，但贝叶斯推断是选择假设最大化后验概率的推断方法。

如果想掌握具体的贝叶斯推断的应用实例，那么可以推荐的书是《贝叶斯预测》。书中基本上没有什么数学公式，很容易阅读。

《贝叶斯预测：关于提高准确率的主观的概率论》

宫谷隆（著）
冈崎裕史（协助计划）
RIC TELECOM 出版
2009年3月
2205日元（含税）
A5规格，192页
ISBN = 978-4-89797-816-1



作为具体的贝叶斯推断的应用例子，书中介绍了用户在移动电话服务公司之间的变更原因分析、犯罪调查、沉船搜索等。

书中还介绍了语音识别所用的隐马尔科夫模型，以及利用朴素贝叶斯来预测未来的销售战略。

模式识别与机器学习

笔者所属的Cybozu Live株式会社，把掌握处理自然语言所必需的机器学习基础知识作为目标，从2011年起，公司内部中谷秀洋先生组织了学习会，会上大家轮流阅读的教材，

就是Bishop先生编写的《模式识别与机器学习》日文版（上下册）。

当时，员工们还不习惯机器学习的思维，读这本教材确实很费神，但是光成滋生先生撰写的相关算法解说，经暗黑通信团出版社装订成《学习模式识别与机器学习》，就成了这本教材的姊妹书。

此书初次出版后，立刻在一些领域成为了话题。在淳久堂书店池袋本店每周销售排名中位于“易读代码”类第一，同一家书店的2012年的计算机书籍销售排行榜名列第三位。

《学习模式识别与机器学习：保证能成功学会贝叶斯理论的数学》

光成滋生（著）
暗黑通信团出版社
2012年7月
1050日元（含税）
B5 规格，99页
ISBN = 978-4873101668



这本书的畅销，也说明大家都在用心学习机器学习，而且还能让初学者感受到，还有和自己一样的大量志同道合的同志。

原书的基本知识是从第2章到第5章，后一半的难关在第9章和第10章。有关第11章的物理学和抽样法，还会有新书出版。

为了学习机器学习的理论，有必要先复习一下大学数学的微积分和线性代数，推荐工程师们一边回忆积分的函数变换和行列式的各种操作，一边学习机器学习。

自然语言处理

在实际数据分析时，因为访问记录中的查找关键字可能是日语或英语，所以需要分析用自然语言写的数据。这时候，学习基本的自然语言处理就变得很重要。这里推荐东北大学的乾·冈崎研究室公开的《语言处理100问题集》。

《语言处理100问题集
—东北大学乾·冈崎研究室》
http：tinyurl.com/gengosyori100

- 第一系列：文本数据的处理基础
- 第二系列：正则表达式，日语的处理
- 第三系列：分割句子，标识化（英语）
- 第四系列：（前半部）查字典
- 第四系列：（后半部）自然语言模型
- 第五系列：词汇解析/描画图形
- 第六系列：段落解析/类
- 第七系列：词汇类似度/聚类分析
- 第八系列：机器学习/分类器
- 第九系列：数据库
- 第十系列：结构化数据（XML）的处理/以CGI为摹本的演示系统

书中从基本课题入手，如在类UNIX系统上从标准输入读取到tab分割数据形式的文本文件处理，同时还涉及Twitter数据分析等实际的编程题目，读者可以循序渐进地学习自然语言的处理技术。

在语言处理的最前沿，也应用了统计的机器学习技术。更进一步的学习教材，推荐高村大也撰写的《为了语言处理的机器学习入门》，此书为触及学习关键的优良教材。

结语

本章介绍的关于学习机器学习、数据分析的技术和理论的教材，都已经介绍完毕。那么什么时候开始学习机器学习呢？现在就开始吧！

Column

机器学习的广泛应用及未来

文/鹿岛久嗣　东京大学　译/雷军

身边日益兴盛的机器学习

粗略地讲，机器学习是数据分析技术的一个流派，但其实它起源于人工智能的一个分支，至今已经有50年的历史。它曾经以各种各样的形式为大家所利用。最近，随着大数据的兴起，对机器学习的注目达到了前所未有的高度（笔者很吃惊地从电视上看到了机器学习的相关词语）。现在，很多企业认为数据分析技术是基础竞争力，因此对其格外重视。

在我们身边，发挥机器学习预测功能巨大作用的实例，就是购物网站的推荐购物功能。该功能被认为是机器学习的主要应用领域，美国一家网上录像出租公司Netflix，几年前曾经投入100万美元奖金举办竞赛征集最优算法。

推荐技术，即通过分析消费者的购物习惯与个人信息等，预测其消费倾向并提示给该消费者。随着数据量的增加，针对每一位消费者的具体情况进行分析，可以提供更加细致的推荐信息。

除网上购物以外，利用社交媒体Twitter或Facebook的朋友想必不少，期间常见的提示信息“这位是不是你的朋友”就是利用推荐技术实现的。推荐技术根据社会学的观点“朋友的朋友还是朋友”，提供有可能成为新朋友的推荐信息。

平时大家在网上浏览时，经常看到的网络广告也是机器学习的应用。如何从众多的广告中挑选最有可能吸引购买的广告提示给浏览者，就是利用机器学习来预测的。

网络广告往往局限于登载篇幅大小以及能显示的次数，为了研究如何达到最大广告效果，需要收集浏览者浏览倾向的信息，同时，根据该信息选择最适合的广告资源。这种技术被称作是强盗预测技术（Bandit），也是一种机器学习的技术。强盗预测技术有着比较长的研究历史，是代替A/B测试，作为网络最优化的强有力方法，引起了广泛的注目。

机器学习在异常检测中的应用

发现式机器学习，在工业中一个非常重要的应用就是异常检测。“异常”就是类似于信息系统中，计算机病毒的感染或者侵入、盗用信用卡、生产线或者机器设备的故障等问题。虽然发生频率不是很高，但是一旦发生就会造成服务停止之类的重大损失，所以在异常发生之前或早期阶段就检测到至关重要。

目前为止，重大事故发生较少，所以根据过去的数据进行预测也是很困难的。相反，既然很难定义异常，那么考虑一下它的反面“正常”是什么。系统正常运转时，产生的数据是有代表意义的，那么掌握正常数据的特征，利用监视系统，一旦发现数据脱离了正常的状态，就发出警告。

这样一来，无须知道出现的是什么异常，只要有可能是异常的时候，立即提示警告，系统管理员就可以继续进行调查。

计算机能超过人类吗？

当今，数据分析技术在各个行业迅速普及和推广，并取得了一定成绩。换句话说，机器的智慧正在威胁着人类的智慧。最近的一件事情，想必大家还记忆犹新，美国IBM的人工智能系统“Watson Computer”十分出色地战胜了国际象棋世界冠军。Watson以自然语言处理技术和语音识别技术为基础，配有庞大的知识储备，加之用机器学习技术开发的核心技术，使其具有应对未知状况的能力。科学家们正计划把这样的系统应用于医疗和金融领域。

这样一来，本来只有人类才能处理的智力性工作，岂不是也会被具有强大大数据处理能力的机器所取代了吗？有些人可能会对此感到不安（同样的担忧，在工业革命时期也曾出现）。其实，这种说法对错参半。正如上述的实例，与人工智能技术的研究范畴相比较，大数据分析技术的世界既是开放和复杂的，又是相当封闭的。那些含有模糊价值和目的、必须超越普通思维的创造性的任务，还是只有人类才能胜任的。

迈向机器和人类共同合作的世界

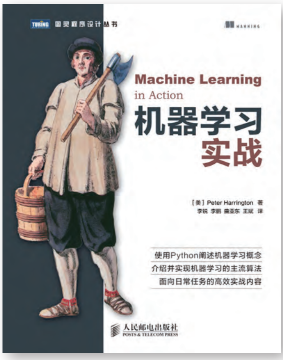
几年前举办的自由式国际象棋比赛（人和计算机可以任意自由组合成团队参加比赛），据说夺冠的团队是象棋业余爱好者和电脑组成的混合团队。这件事情提示了人们，今后的趋势将是机器和人类的共同合作。也就是说，不是机器和人类中的某一方，而是人类和机器互相弥补才能发挥出最佳的表现力。

实际上，这正是近年来计算机科学界引人注目的领域——人计算（human computation）。对媒体的理解或搜索等，虽然可以实现部分程度的机械性操作，但其中很难实现部分还是必须依靠人来解决的。这部分实际上就是在计算机系统里采用了人类明确的指令，来实现单靠机器不能完成的功能而已。为了实现实时性，全世界的人利用网上众包等形式可以实时访问这些资源。

回到数据分析的话题上，实际上，数据分析是机器学习中自动分析的一部分而已。例如理解业务、设定课题、数据的收集和整理、结果的解释等，数据分析的很多部分，都是劳动密集型的专门性处理工作，那么，就像人计算一样，人类与机器相互取长补短才能创造出更高的价值。难道这不正是未来的数据分析方式吗？

TURING 图灵教育 iTuring.cn

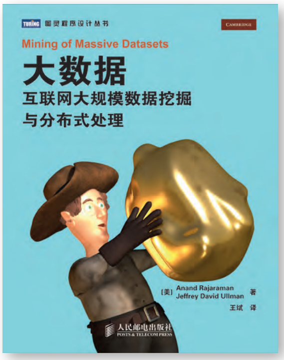
【推荐阅读】机器学习和数据分析



图灵访谈 Peter Harrington：如何成为一位数据科学家

作者：Peter Harrington
译者：李锐 李鹏 等
书号：978-7-115-31795-7
定价：69.00 元 /39.99 元
开本：16
页数：332

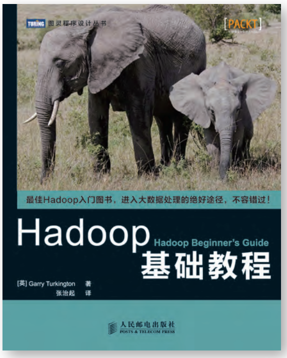
- 使用 Python 阐述机器学习概念
- 介绍并实现机器学习的主流算法
- 面向日常任务的高效实战内容



图灵访谈 Jeffery Ullman：大数据不是噱头，值得投入我们的力量

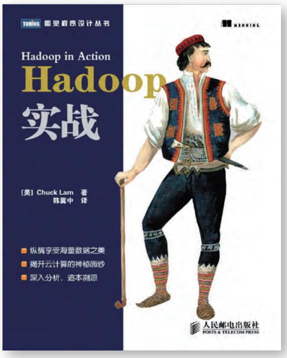
作者：Anand Rajaraman 等
译者：王斌
书号：978-7-115-29131-8
定价：59.00 元
开本：16
页数：272

- 大数据时代的及时雨
- 全球著名数据库技术专家最新力作
- 理论与实际算法实现并重



作者：Garry Turkington
译者：张治起
书号：978-7-115-34133-4
定价：65.00 元 /39.99 元
开本：16
页数：324

本书揭开了 Hadoop 的神秘面纱，它着重讲解了如何应用 Hadoop 和相关技术搭建工作系统并完成任务。在读者明白这些内容之后，又介绍了如何使用云服务完成相同任务。本书从 Hadoop 的基本概念和初始设置入手，讲述了如何开发 Hadoop 程序，如何在数据规模增长的时候维持系统运行，本书涵盖了有效使用 Hadoop 处理实际问题所需用到的全部知识。



图灵访谈 韩冀中：浅谈 Hadoop 实战

作者：Chuck Lam
译者：韩冀中
书号：978-7-115-26448-0
定价：59.00 元
开本：16
页数：268

本书旨在教会你如何以 MapReduce 方式编写程序。书中内容由浅入深，以几个简单的例子开始，继而转向 Hadoop 在较为复杂的数据分析中的应用。其中包含 MapReduce 编程中的最佳实践及设计模式，还介绍了 Streaming API 及 Pig 和 Hive 等工具。



作者：徐立冰
书号：978-7-115-31150-4
定价：59.00 元
开本：16
页数：312

通过阅读本书，读者将清楚地了解到如何在云计算与大数据时代构建安全、可靠、高速与灵活的网络。本书主要内容包括：云计算对基础架构的驱动、云计算网络的组成、如何构建安全可靠灵活的网络通道、虚拟化数据中心的扩张、外部和内部网络的实现、大数据网络设计要点，以及厂商解决方案等。



作者：城田真琴
译者：周自恒
书号：978-7-115-31787-2
定价：49.00 元
开本：16
页数：268

本书是日本畅销的大数据商业应用指南。书中结合野村综合研究所独家披露的调查数据，网罗了美国、日本标杆企业与政府的应用案例，总结了大数据的商业模式，以及在大数据应用中需要注意的隐私问题，并就如何为大数据时代做好准备展开了深入的探讨，提出了诸多有益的建议。

本书适合商业人士以及与大数据相关的 IT 从业者阅读。

特辑 2

原因

是否拥有自己的评价标准？

能否发现系统性能瓶颈？

基准测试应用技术

购买了新型服务器，投入使用后发现性能并不理想的情况时而有之吧？参考产品手册的机器性能介绍来选择想要购买的机器时，首先就要进行基准测试。在本特辑中，我们以日常使用的PC为例，通过了解硬件体系结构来对性能瓶颈及其原理进行探讨学习。同时了解服务器如何在网络环境中发挥作用，并通过对各种基准测试方法的学习，掌握硬件评价的技巧，最后总结出一套通用的评价标准。

邮电

CONTENTS



Part 1

基准测试的基础 [PC篇]

第一部分主要讲解PC基准测试的必要性与方法。现今，多数基准测试的操作都非常简单，点击一下即可进行一系列的数据处理，最后得到结果。然而，如果能够深入地理解其中的含义，就可以判断测试结果是否符合要求，从而更加有效地应用这些数据。

文/圆藤优沙 Twitter @YuusaEndo 译/叶东

基准测试的必要性

有读者可能对“基准测试”这个词早有耳闻。基准测试主要是对计算机的处理速度进行测试，即让测试程序在计算机上运行，掌握其运行速度和其他性能指标。

之前你可能听说过某某超级计算机跻身世界50强等。就超级计算机而言，基准测试程序LINPAK[①]测试的速度是决定世界排名的标准。

基准测试不仅可用于Intel系列硬件，也适用于整台计算机。测试结果出来后，就可得到此种计算机相对于通用基准值的高低状况。

有些人认为，为了进军超级计算机500强而进行基准测试是可以理解的，但一般的团体、企业则没有必要进行基准测试。这种说法过于片面，因为实际上基准测试有利于业务处理速度的提升和成本的降低。本文的宗旨正是通过基准测试将有限的资源得到最大程度的应用。

① http://ja.wikipedia.org/wiki/LINPACK 或者 http://top500.org/project/linpack/

应在什么时候进行基准测试?

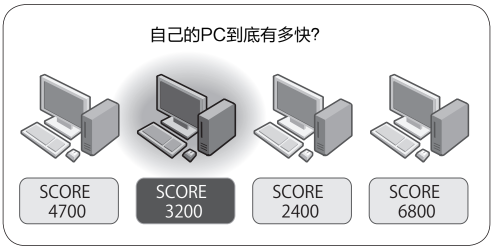
如果只看基准测试的结果没有任何意义，只有将它与通用的基准值进行比较才能看出孰优孰劣。一般有以下几种情况会用到基准测试。

情况1：将某台计算机的性能与周边的计算机作比较

如图1所示，PC硬件的升级与软件的复杂程度互相影响。即使现在感觉跑得很快的机器，几年后伴随软件的更新，也会变得愈发迟缓。这时可以通过基准测试来量化指标，确定是继续使用还是更新换代。

简而言之，就是用基准测试将快与慢的感

▼ 图1　将某台计算机的性能与周边的计算机做比较



觉量化。感觉机器慢了就需要更换，这样太过主观。如果能够用数据来说话，就更加具有说服力。

情况2：通过基准测试来提高计算机性能，进而确保运行的稳定性和速度

基准测试的另一个目的是找到机器的瓶颈，权衡性能与成本，对其进行改善。

例如，服务器需要承受激增的交易服务。工作站需要减少模拟计算所花费时间。而这些都很难使用业务软件进行测试与评估，因为环境的配置与类似情况的再现是非常困难的。因此，使用通用测试程序进行测试与评估变得非常有意义。

理解基准测试结果所需要的预备知识

一般来讲，基准测试程序都是利用多个函数进行组合运算，最终得到一个综合的结果。要想更好地把握结果，需要详细了解从开始到最终得出结果的过程。下面，先讲讲PC大致的数据处理流程。

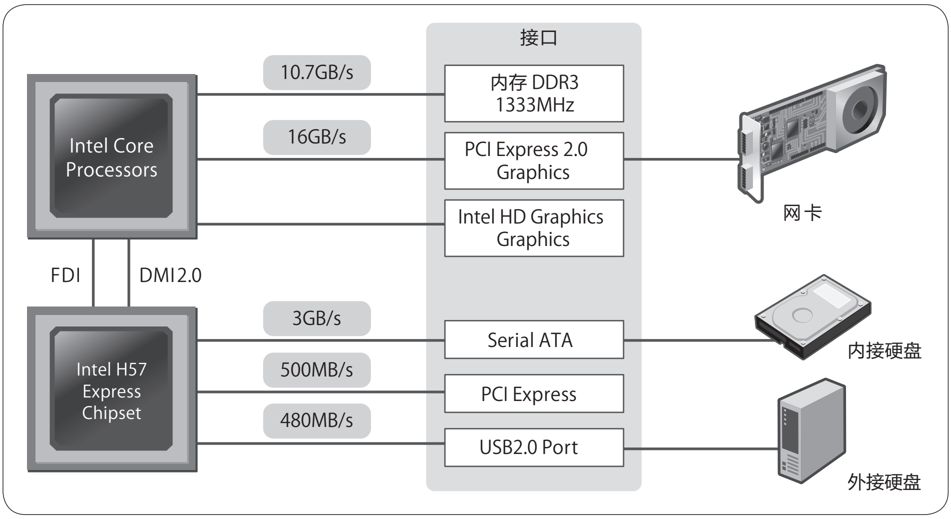
传输速度的确认

Intel的Core i系列各种总线的传输速度如图2所示。各接口的传输速度是根据Intel资料取得的，而实际接线情况应该更多，可以更加高效传递数据，例如存储器交叉存取。应该理解，这一结果与实际系统内数据的传输速度还是存在一定差异。

依靠CPU缓存中的命令，数据通过I/O管理芯片传送到硬盘等存储装置上。硬盘上的数据根据命令被送往CPU，作为图像处理的结果被输出。这一系列的处理经过了各种各样速率不同的总线。另外，各种装置自身读与写的速率也不同，进而影响整体处理速度。

所以，PC性能是的下面几点相互作用的结果。

▼图2　计算机的结构图与传输速率指标（以Intel H57 Express芯片为例）



- CPU处理能力的高低
- 各种总线传输速率的高低
- 各种外设装置的处理速度（存储装置的读写操作等）

这些只是数据处理的基本流程。还存在其他影响机器性能的重要因素，例如数据的大小、处理命令优先级，以及各种外设装置的使用频度等。

处理变慢的原因是什么？

处理大于CPU内部缓存的数据时，需要使用存储器完成。存储器容量虽然比内部缓存大，但是传输速率与存取速率要劣于内部缓存。如果遇到存储器也不能容纳的数据，溢出部分就需要用硬盘的虚拟存储区域来完成。这样，低速外设的读写处理就造成性能的降低。虚拟存储区域如果使用的是外接式硬盘的话，因为外接式硬盘需一般通过USB方式读与写，因而性能会进一步降低。上述内容是以使用虚拟存储区域为前提而言的。

计算机性能的高低与使用方法和组成结构息息相关，所以以实际应用的负荷状态对机器各种指标进行基准测试是非常重要的。如果已知某项性能需要改善，就等于确立了目标。举一个流体分析的例子：程序从一个状态计算完成后，对于状态的变化还需要进行再计算。类似这样的反复处理是必要的（图3）。在这种情况下进行基准测试，可能由于硬盘处理速度过缓而造成了系统瓶颈。作为解决方案，可将硬盘组建为阵列模式以提升读写速度。

基准测试的分类与注意事项

基准测试所使用的程序有许多种，本文着重介绍以下三种。

● **综合型基准测试**

对于由CPU、存储器、显卡等组成PC的所有组件进行整体基准测试。

● **结构型基准测试**

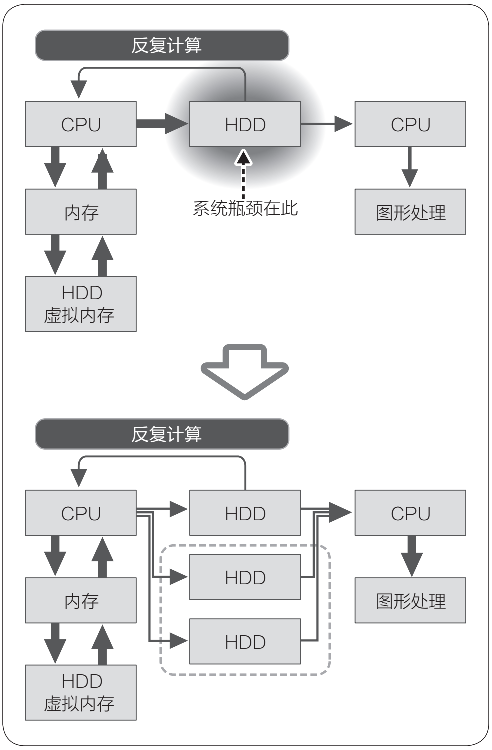
针对存储器、显卡等个别组件进行基准测试。

● **特定目的型基准测试**

在与实际应用相接近的条件下进行基准测试。

我们应根据测试对象和测试目的，来选择

▼ 图3　反复计算示意图



相应的基准测试程序。例如，DirectX、OpenGL等显卡评价程序，主要用于对游戏、CAD等环境的测试，而对于一般服务器则是不需要的。应用于事务处理的MS Office环境，对于浮点小数演算方面的测试是不必要的。

另外，还需要注意的是，有些基准测试程序未对应64位机器，或者未对应多核CPU，使用前应该仔细阅读相关资料。

尝试一下基准测试

下面我们来介绍各种基准测试工具以及使用方法（大部分都是基于Windows版软件）。

综合型基准测试

◆ PCMark 7

PCMark 7（图4）由Futuremark公司开发，为PC版应用程序。可以完成综合型基准测试。现在有3个版本，最基础版本是Basic Edition，为非商用免费；能够支持更加复杂测试的版本是Advanced Edition；更高级的就是可以执行命令行程序的Professional Edition（图5）。使用Basic Edition可用于测试Video playback and transcoding、Web Browsing、DirectX 9、System Storage等各项指标以及综合指标。Advanced Edition除了基本测试项目之外，还具备Common usage performance分类测试功能，Computation和System storage可以取得PC各组件的基准测试结果，并且按照用途进行测试（表1和图6）。

◆ 如何分析测试结果

使用PCMark进行基准测试的结果可与网站上公布的其他系统的测试结果做比较。PCMark软件的view Result on PCMark.com按键可以用来显示自身测试结果。如果需要与其他系统作比较，选择网页的Results指定对象就可以直观对比。

▼ 图4　PCMark 视频测试中的截屏



图7是一个做对比的例子，在CPU性能基本相同的情况下，总体性能不分上下，但是能够看出作者这台机器的显卡性能不是太高。

与不同种CPU（Intel Core i7）的最高性能结果相比，显卡（DirectX 9）性能差距是9倍，web browsing的性能差距是20倍，遭到不小的打击。

▼ 图5　Basic Edition 只用于测试PCMark Score（上）、Advanced Edition的详细显示结果（下）





然而，如果没有使用DirectX玩游戏，也没有感觉网速过慢的话，也就没有什么大不了的了。

结构型基准测试

结构型基准测试的产品基本都是与显卡相关联的。代表性的显卡基准测试工具有OpenGL和DirectX。OpenGL是开源的多平台型显卡库，主要用于CAD、CG等方面的测试。DirectX原本是用于Microsoft Windows游戏软件的开发，最近也能用于CAD等方面的测试。

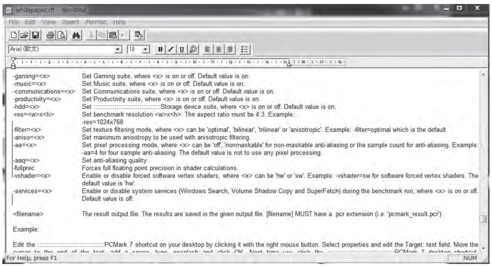
◆ OpenGL基准测试工具介绍

MAXON的CINEBENCH（图9）是OpenGL针对显卡和CPU进行基准测试的工具。支持Windows和Mac OS X。按下CINEBENCH键后，程序自动执行，非常简便。测试结果在窗口左面的排行榜中显示。CINEBENCH的测试结果常会被公布在很多PC评

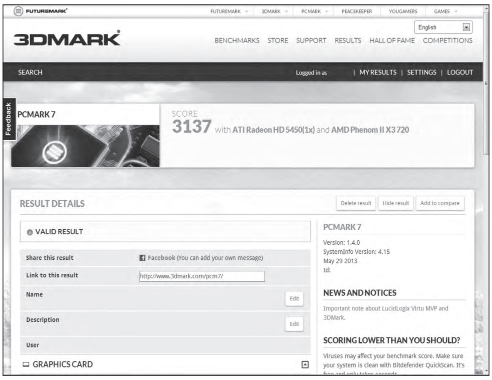
表1　PCMark7的基准测试项目（评价项目以分组形式构成）

PCMark suite	数值运算、存储装置、显示等综合测试
Productivity suite	Text Editing, Web Browsing and decrypting, System storage等办公处理类功能测试
Entertainment test	Video playback and transcoding, DirectX 10等娱乐类功能测试
Creativity suite	High Quality Video transcoding, Image manipulation等图像处理关系的测试
Computation suite	Video transcoding, Image manipulation的测试
System Storage suite	各种数据（picture, video, music, gaming等）的输入输出测试

▼ 图6　Text Editing的基准测试结果



▼ 图7　PCMark的结果被记录在网页

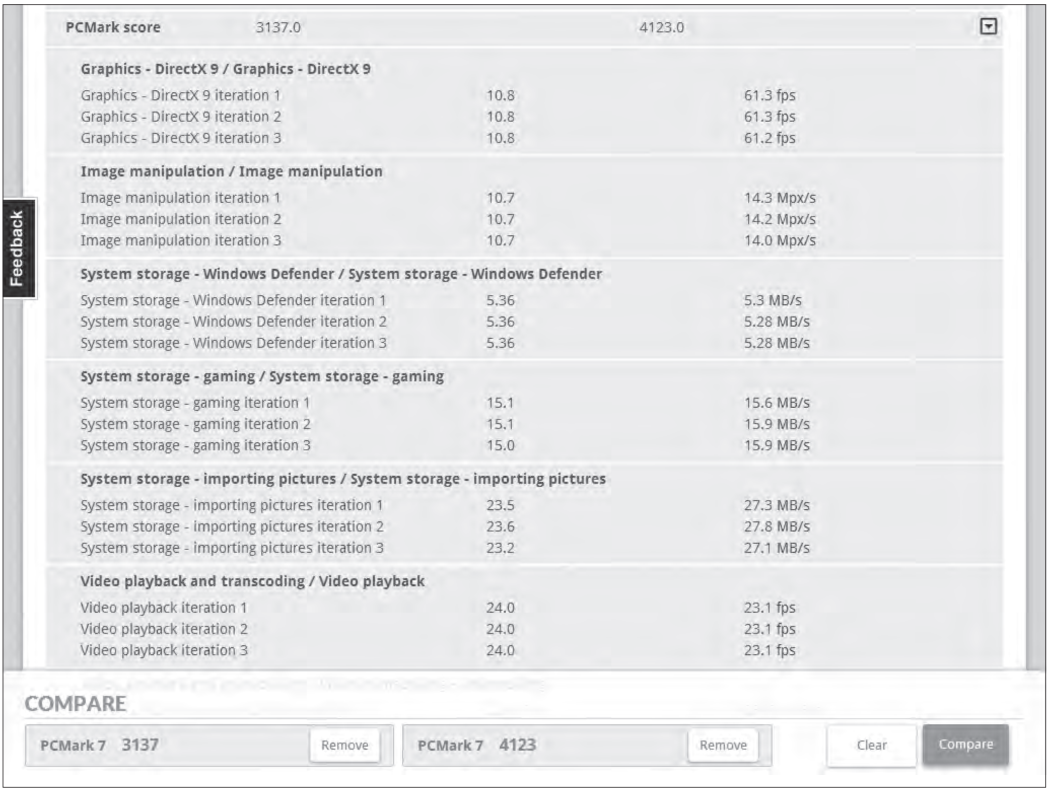


价杂志与网站上。CINEBENCH既可以测试多核CPU，也可以测试单核CPU。可以测定程序运行在多核状态与单核状态性能的差异，并且支持命令行启动。

◆ DirectX基准测试工具介绍

与CINEBENCH同样，Futuremark的3DMark（图10）作为DirectX基准测试工具被PC杂志和网站广为介绍。本文执笔时，已经有与DirectX10相对应的3DMark VANTAGE，以及和DirectX11相对应的3DMark11。并且，公司还发布PCMark的免费版（Basic Edition），和可以支持自定义设置的Advanced Edition以及可以支持命令行执行的Professional Edition（可用于演示的无限循环处理功能）。它们大部分是针对于性能需

▼ 图8　自己系统的内核信息，可与网页上公布的其他系统进行比较。



PCMark score	3137.0	4123.0
Graphics - DirectX 9 / Graphics - DirectX 9		
Graphics - DirectX 9 iteration 1	10.8	61.3 fps
Graphics - DirectX 9 iteration 2	10.8	61.3 fps
Graphics - DirectX 9 iteration 3	10.8	61.2 fps
Image manipulation / Image manipulation		
Image manipulation iteration 1	10.7	14.3 Mpx/s
Image manipulation iteration 2	10.7	14.2 Mpx/s
Image manipulation iteration 3	10.7	14.0 Mpx/s
System storage - Windows Defender / System storage - Windows Defender		
System storage - Windows Defender iteration 1	5.36	5.3 MB/s
System storage - Windows Defender iteration 2	5.36	5.28 MB/s
System storage - Windows Defender iteration 3	5.36	5.28 MB/s
System storage - gaming / System storage - gaming		
System storage - gaming iteration 1	15.1	15.6 MB/s
System storage - gaming iteration 2	15.1	15.9 MB/s
System storage - gaming iteration 3	15.0	15.9 MB/s
System storage - importing pictures / System storage - importing pictures		
System storage - importing pictures iteration 1	23.5	27.3 MB/s
System storage - importing pictures iteration 2	23.6	27.8 MB/s
System storage - importing pictures iteration 3	23.2	27.1 MB/s
Video playback and transcoding / Video playback		
Video playback iteration 1	24.0	23.1 fps
Video playback iteration 2	24.0	23.1 fps
Video playback iteration 3	24.0	23.1 fps

COMPARE

PCMark 7　3137　Remove　PCMark 7　4123　Remove　Clear　Compare

求很高的游戏而开发的。3DMark与PCMark同样，基准测试结果数据公布在很多网站上（图11）。

◆ 存储器基准测试工具介绍

硬盘与闪存等存储器的性能因素对PC整体性能有莫大的影响，这个前面曾经提到过。Crystal Diskmark（图12）可以免费且方便地评测出存储器的性能。它支持线性读写处理、可变长数据随机读写、测定结果的实时显示等功能，数据结果客观且易于理解。而且，测试用文件大小可以随时变化，这是为存储器拥有大容量缓存，而数据文件又比较小的情况下，能够正确测量缓存性能而准备的。在此之前，对于测试时缓存溢出的情况，往往会出现写操作性能极端降低的情况。而利用这一功能，通过改变文件的大小则可以解决这一问题。

▼ 图9　MAXON CINEBENCH（OpenGL基准测试工具）



▼ 图10　Futuremark 3DMark（Direct X基准测试工具）



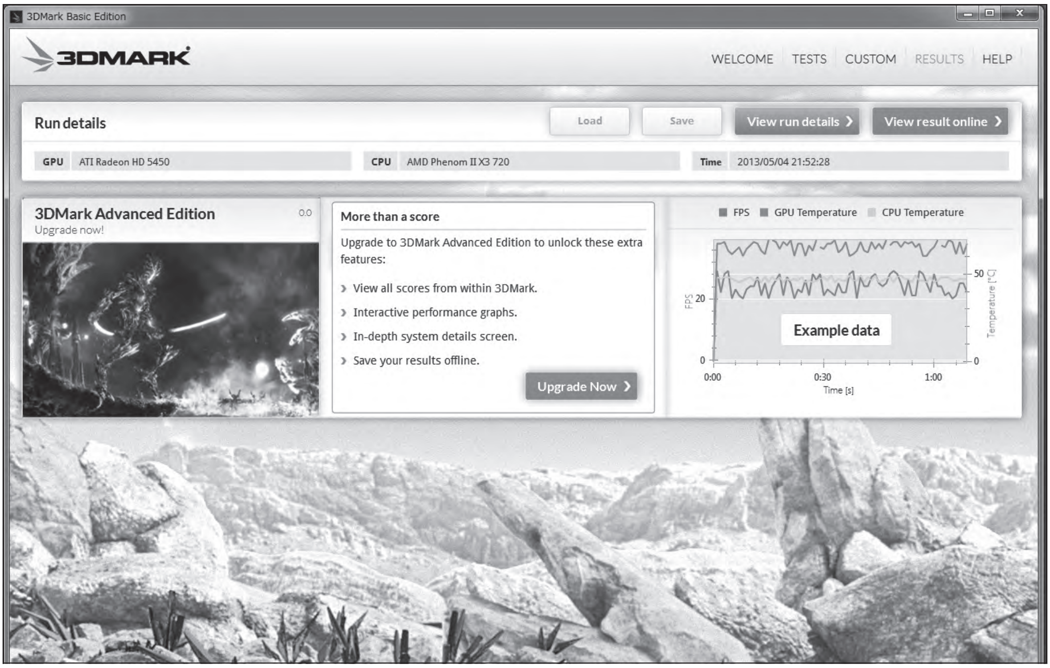
应用程序式的基准测试工具介绍

◆ 姬野基准测试工具（himenoBMTxp）

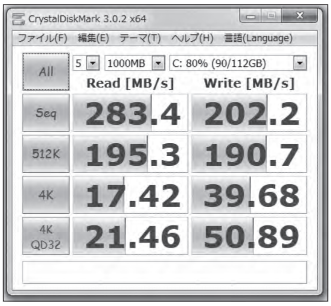
一般的PC是以整数运算为主，用于模拟流体分析的程序则是采用浮点小数运算，从一个状态得到结果进行再计算，循环往复地进行。这种计算往往需要上万回，因此运算速度是至关重要的。姬野基准测试工具是面向理化研究所的计算中心而制作的非压缩流体分析型基准测试工具（图13）。

这种测试工具的计算结果以FLOPS（Floating-point Operations Per Second）为单位，它表示1秒内浮点小数运算的次数。除了Windows版和Mac版，姬野还支持在FORTRAN语言和C语言的开源程序，使用很广泛。

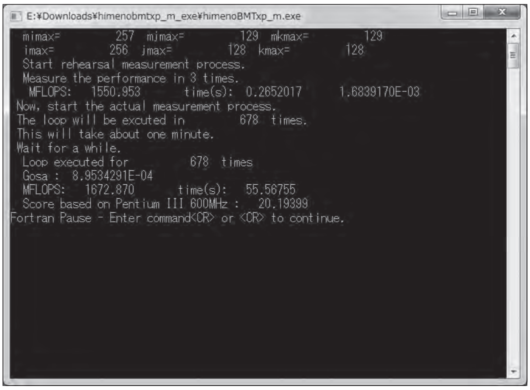
▼ 图11　3Dmark内核信息在Web网页的显示



▼ 图12　CrystalDiskmark



▼ 图13　姬野基准测试工具

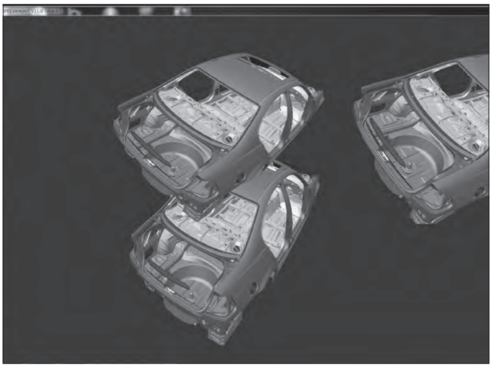


◆ SPECviewperf

这是SPEC（Standard Performance Evaluation Corporation）公司面向OpenGL的基准测试工具（图14）。它最大的特征是其主体由一些软件供应商直接提供的产品组成，可以用来测试实际运用软件的执行情况，得到性能结果。显卡的测试结果单位是frames/sec，保存在/SPECv/SPECviewperf/viewperf/viewperf11.0/results这个路径下，且针对CAD、CG的测试提供了Viewset功能（参照下图）。

*() 内为Viewset名称
LightWave(Light01)
CATIA(catia-03)
Ensight(ensight-04)
Maya(maya-03)
Pro/ENGINEER(proe-05)
SolidWorks(sw-03)
Siemens Teamcenter Visualization Mockup (tcvis-02)
Siemens NX(snx-01)

▼ 图14　SPECviewperf



其他基准测试工具

最纯粹的基准测试的工具是为对象软件量身定做的。例如非常有名的模拟软件ANSYS Fluent（图15）。它提供了用于基准测试的代码[②]。这样，测试结果就可以和实际结果直接进行比较，从而得到性能的比较数据。

◆ TPC-C

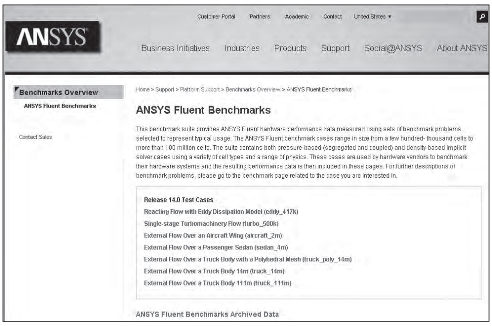
TPC-C是Transaction Processing Performance Counci l[③]的产品，主要用于服务器的交易处理性能测试，亦可用于PC与数据库的基准测试。这里就不多介绍。

结　语

本节简单介绍了基准测试的基础知识。如果想要深入了解还需要学习更加专业的知识。希望这些知识能对你有效利用PC知识有所帮助。

② http://www.ansys.com/Support/Platform+Support/Benchmarks+Overview/ANSYS+Fluent+Benchmarks
③ http://www.tpc.org/tpcc/

▼ 图15　参考ANSYS Fluent的网页



Part 2

基准测试的基础 [服务器篇]

第二部分主要讲解关于服务器基准测试的内容。包括基准测试的项目组成、测试目的与执行计划，以及需要特别注意的地方。同时会对服务器的构成作进一步说明。

文/藤城拓哉　Twitter @tafujish　译/叶东

影响服务器性能的关键因素

服务器由CPU、存储器、硬盘、网卡等组件构成，在购买前或使用时需要考虑配置的级别。服务器的性能是必须要考虑的关键因素。

影响服务器性能的最大因素是CPU。CPU的主要指标是内核数与主频。仅次于CPU的因素是内存与硬盘。一般在选购的时候，可能只会想到需要的容量，但也要把处理性能考虑在内。特别是硬盘，如果其处理速度低于其他组件，很容易成为系统瓶颈。

如何了解处理性能

那么我们应该如果了解各组件的处理性能呢？

首先，可以在商家的主页上获取相关信息。在网站上有可能查到所有产品的参数列表，还可能查到在理论峰值和理想环境下运行的性能指数。这些信息作为参考资料是没有问题的，可是在实际运行环境下得到的数值会存在一定偏差。

要了解实际运行环境下的性能指数，只能自己动手进行基准测。

需要哪些基准测试

在做基准测试之前，先要确定执行哪些内容，即需要取得哪些方面的信息。

首先，需要取得CPU、硬盘等组件的基准测试信息。然后根据这些信息再去测试其他组件。

其次，需要安装中间件，利用它来获取基准测试的信息。例如，通过Apache httpd可以得到HTTP服务器的基准测试信息，通过MySQL可以得到数据库服务器的基准测试信息。它们获得的数据应该都非常接近实际运用的状态。

最后，对目标应用或系统进行负载测试，以验证性能是否达到理想状态，确认哪里还存在系统瓶颈。

倘若在测试开始就已有目标应用，那就会比较方便。但在选定服务器时，目标应用往往还未最后完成，这样就只能通过对各个组件和中间件进行基准测试，来模拟实际运行指标。

对于产生系统瓶颈的原因，需要深入地调查，这时就需要了解各组件的关联性。例如，

就数据库服务器而言，CPU与硬盘存在很深的相互依存关系，接下来我们就看一看CPU、硬盘、内存和网络接口的基准测试结果吧。

进行基准测试前需要注意的地方

明确目的

首先要明确基准测试的目的。例如，对于服务器升级而言，新服务器与旧服务器相比，性能提升多少个百分点？配置新的服务器时选择哪些厂商的产品？这样就需要比较若干服务器或CPU的基准测试结果。具体来说，应把与测试的关联项目整理出来以后，再进行基准测试。

举一个其他的例子，比如对于耗电量与性能比、“单个机柜耗电上限为4千瓦，能得到的最大性能是多少”等关于耗电量的问题，不仅要进行基准测试，还要测量耗电量。

另外，在目标应用程序尚未完成的情况下，应让操作系统和中间件尽可能接近实际运行环境。如果使用与目标应用程序功能类似的基准测试程序，就能得到更加接近实际运行环境的测试结果。

综上所述，漫无目的地进行基准测试是不可取的，需要事先计划，以缩短测试时间，防止遗漏测试项目。

检查BIOS的设置

基准测试实施前，需要检查服务器BIOS的设置。

特别是省电功能，对整体性能的影响是很大的。在服务器上我们经常能看到“高耗电时性能最高”、“最高性能”、“耗电优先”等选项（图1）。当需要测试最大性能的时候，需要选择“最高性能”。缺省值的设置方法，各厂家都不一样，需要注意。

确保在相同环境下测试

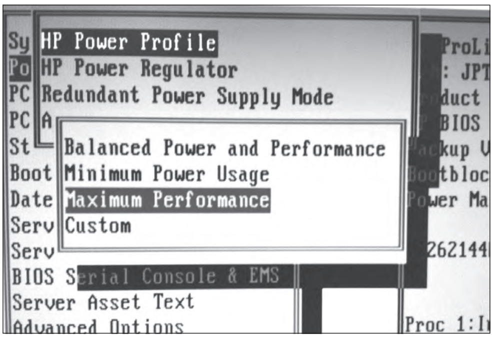
多项目的基准测试应该尽量在同一环境下实施。也就是说，基准测试工具、操作系统、编译器（GCC等）的运行版本要一致，以防版本差异造成测试结果的偏差。

另外在基准测试中，需要尽量降低不必要的机器负荷。对于Linux、UNIX等操作系统，利用ps或top等命令就可以关闭不必要的进程，以减轻机器负荷，从而保证测试结果的准确性。

检测系统瓶颈

一般来说，基准测试工具在获取实际数据前需要反复测试，检查是否存在预料之外的系统瓶颈，以及预料中的系统瓶颈部分是否真的超负荷。例如，基准测试工具自身是否给CPU

▼ 图1　BIOS设置的参考示例



造成过高负荷，或者缓存中的数据无法通过I/O正常到达存储装置等。另外，对于客户端/服务器型的系统来讲，客户端也是可能产生瓶颈的。

另外还需注意物理的系统瓶颈。例如，如果网络基准测试对象的物理带宽是1 Gbit/s，那么它的性能上限就是1 Gbit/s。同样，硬盘的SATA接口、SAS接口或PCI总线接口都存在数据传送速率的上限。虽然这些地方基本上不会出现系统瓶颈，但了解一下基准测试对象的物理构成也不是坏事。

虚拟环境的基准测试

对虚拟环境做基准测试是比较困难的。基准测试工具的基本流程是，在进行一定量的处理后，测试完成此处理所花费的时间，或者在单位时间可以完成处理的数量。也就是说，处理花费了多长时间是至关重要的。然而，虚拟机环境中测出的时间往往是不准确的。虚拟环境并不是对CPU进行实时处理，时钟单元也经常处于模拟状态。系统管理程序（hypervisor）的种类和版本不同，对CUP负载测试的影响也不同。另外，其他虚拟机所产生的负荷影响也是存在的。

作为解决方案，因为客户端/服务器型的系统是适合进行基准测试的，所以可以把物理机作为客户端，虚拟机作为服务器，这样就能降低虚拟环境造成的影响了。

不过可能有些人希望能够使用自己常用基准测试工具。那就需要事先进行扎实的测试，确认测试时间是否有较大的出入。因为处理时间是作为结果输出的，所以如果有出入应该能很容易察觉出来。当然，也可以通过拉长运行时间来降低误差。在过去，当时间出入较大时，测试人员甚至还亲自用秒表来计时。

综上所述，对虚拟环境做基准测试需要精心地准备，结果的正确与否也需要花功夫去核实。

CPU的基准测试

接下来进入实践部分，我们一起来对CPU进行基准测试。服务器的配置如表1所示，之后所有的测试都是在这个服务器上进行的。

什么是CPU内核数？什么又是主频？

从CPU的性能来看，内核数、主频应该重视哪个呢？众所周知，所有软件基本上都是主频越高，运行的速率越快。但并非内核数越多，

表1　服务器的配置

机器种类	HP ProLiant DL360p Gen8
CPU	Intel Xeon E5-2670（2.6GHz/8核）2个
Memory	PC3-10600R 8GB 4组
Disk	SAS 15000rpm RAID10 4组，RAID缓存1GB
NIC	Intel X520-DA2 10 Gbit/s
OS	CentOS 6.4 x86_64（最小安装），使用系统标准GCC

运行的速率越快。因为多核速率的提高，还取决于并行处理的个数。例如，一个拥有8核的CPU上只运行了4个进程，这就说明只使用了CPU性能的一半。使用Apache httpd生成多个进程、在同一服务器上运行多个软件，或者使用多线程处理应用，都可以体会到多核的好处。

进行基准测试时，还需要注意并行处理的个数问题。一般来说，非并行处理就相当于单核测试。如果是多核测试，则应该增加并行处理的个数。

多核的基准测试还有一点比较重要。Intel公司的Hyper-Threading Technology（以下简称HTT）产品，把一个物理内核看作两个逻辑内核，虽然可以当作双核CPU，但是性能上还达不到加倍的效果，最多只能提升10%～30%。使用的软件不同，性能的提升率也不一样，即便用同样的基准测试程序来测试，结果也会产生差异。所以，HTT设置为有效还是无效，对于测试结果也会产生至关重要的影响。

综上所述，对于CPU基准测试，需要注意针对的是单核还是多核。

什么是CPU性能指标的单位

CPU性能指标的基本单位是MIPS（Million Instructions Per Second） 和 FLOPS（Floating-point Operations Per Second）。MIPS是1秒钟可以执行命令的数量（100万次），FLOPS是1秒钟内浮点小数的运算次数。浮点小数计算在HPC（High Performance Computing）中被广泛使用，但对于一般的服务器来讲并不重要。

我们来简单地看一看MIPS数值，执行下面的命令。

```
$ cat /proc/cpuinfo
~省略~
bogomips  : 5187.33
~省略~
```

我们得到了以内核为单位的BogoMips的结果。数值越大代表处理速度越快。伴随主频的提高，这个数值也会变大。但是，Bogo也正和bogus（假货）的意思一样，只是单纯地重复执行命令，对CPU性能的表达并不全面。

下面就来介绍一下真正的基准测试工具。

UnixBench

UnixBench[①]是一款很老的基准测试工具，是针对Dhrystone[②]、Whetstone[③]等比较传统的CPU制作的。它通过文件复制处理、SHELL操作，以及进程生成、系统调用、超负荷运行等与操作系统相关的部分的测试得到结果。UnixBench不只用于CPU的基准测试。

◆ UnixBench的编译与运行

如图2所示，首先需要安装编译程序包，然后下载UnixBench源程序，通过编译后就可以运行了。如果项目的测试时间很长，则只测试3次，如果短，那么可以运行10次。先运行一次“并行数为1”的测试，然后执行“并行数为内核数”的测试。完成全部测试需要一个小时左右。

如果目标服务器的理论内核数超过16，就

① http://code.google.com/p/byte-unixbench/
② 由Reinhold P.Weicker在1984年开发的不包含浮点小数运算、面向整数和文字运算的性能测试程序。
③ 英国国家物理研究所开发的以浮点小数运算为主的性能测试程序。

不能直接进行并行测试。实在需要的话，请参考UnixBench网页上题为Can't do default run completely with > 16 CPUs的帖子，那里介绍了使用批处理程序实现的方法。

◆ UnixBench测试结果

测试结果和标准输出一起以文本和html的形式保存在results目录中。测试结果由"并行数为1"和"并行数为内核数"组成。

笔者运行了"并行数为32"的测试，结果请参考图3。最后可得出System Benchmarks Index Score的数值为8613.1的结果。每个测试的结果，都是以Sun SPARCstation 20 SM61（1995年的计算机）的性能基准值为10进行比较计算的。

如果需要系统的综合性能指标，可以参考这个Score数值。如果需要CPU的整数计算性能，可以参考Dhrystone的测试结果。

▼图2　UnixBench的编译和运行

```
$ sudo yum -y install perl perl-Time-HiRes make gcc   ←下载软件包
$ curl http://byte-unixbench.googlecode.com/files/UnixBench5.1.3.tgz | tar zx   ←下载工具
$ cd UnixBench
$ ./Run   ←运行编译与基准测试的程序
```

▼图3　UnixBench的运行结果

```
32 CPUs in system; running 32 parallel copies of tests

Dhrystone 2 using register variables      579640366.1 lps   (10.0 s, 7 samples)
Double-Precision Whetstone                    57707.7 MWIPS (10.0 s, 7 samples)
Execl Throughput                              50979.7 lps   (29.9 s, 2 samples)
File Copy 1024 bufsize 2000 maxblocks        583796.7 KBps  (30.0 s, 2 samples)
File Copy 256 bufsize 500 maxblocks          167663.0 KBps  (30.0 s, 2 samples)
File Copy 4096 bufsize 8000 maxblocks       1829440.9 KBps  (30.0 s, 2 samples)  ←各项测试的结果
Pipe Throughput                            27739873.2 lps   (10.0 s, 7 samples)
Pipe-based Context Switching                6562696.1 lps   (10.0 s, 7 samples)
Process Creation                              95763.8 lps   (30.0 s, 2 samples)
Shell Scripts (1 concurrent)                  99406.3 lpm   (60.0 s, 2 samples)
Shell Scripts (8 concurrent)                  14489.4 lpm   (60.1 s, 2 samples)
System Call Overhead                        5456787.3 lps   (10.0 s, 7 samples)

System Benchmarks Index Values               BASELINE       RESULT    INDEX
Dhrystone 2 using register variables         116700.0  579640366.1  49669.3
Double-Precision Whetstone                       55.0      57707.7  10492.3
Execl Throughput                                 43.0      50979.7  11855.7
File Copy 1024 bufsize 2000 maxblocks          3960.0     583796.7   1474.2
File Copy 256 bufsize 500 maxblocks            1655.0     167663.0   1013.1
File Copy 4096 bufsize 8000 maxblocks          5800.0    1829440.9   3154.2
Pipe Throughput                               12440.0   27739873.2  22298.9
Pipe-based Context Switching                   4000.0    6562696.1  16406.7
Process Creation                                126.0      95763.8   7600.3
Shell Scripts (1 concurrent)                     42.4      99406.3  23444.9
Shell Scripts (8 concurrent)                      6.0      14489.4  24149.0
System Call Overhead                          15000.0    5456787.3   3637.9
                                                                   ========
System Benchmarks Index Score                                        8613.1  ←最终的运行结果
```

本次的测试条件为8核2组的新型CPU，将HTT设置为有效，最后得到了高效的处理结果。

在HTT无效的情况下，Score的数值为7412.0，比HTT有效时处理速度降低了20%。

硬盘的基准测试

以下进行硬盘I/O的基准测试。这种测试适用于硬盘与硬盘组的RAID阵列或外置存储器。

硬盘I/O的基准测试

基准测试对象有几种组合方式。例如，SATA硬盘、SAS硬盘或者回转速度（RPM）都会对速度产生影响。而对RAID阵列来说，硬盘数和RAID级别对速度也会产生一定影响。RAID控制卡也是影响因素。外部存储器也是同样有硬盘数和RAID级别，控制器的性能也会对速度产生影响。

无论是怎样的结构，都必须是操作系统可以识别的设备。但是，作为文件系统，都需要格式化和设备链接等操作，所以文件系统也是影响设备I/O处理速度的因素。

硬盘I/O的处理有4种情况。

- 读处理（Read）、写处理（Write）
- 线性I/O处理、随机（无序）I/O处理

用于大文件的处理和大量日志输出的时候，一般采用线性I/O处理。而用于大量小文件的处理以及数据库操作的时候，一般采用随机（无序）I/O处理。

◆ 硬盘的I/O处理指标

硬盘的I/O处理指标，主要是传输量、IOPS（Input/Output Per Second）、应答延迟率等。传输量（也叫带宽）是1秒钟传送数据的量，单位是Byte/sec（B/s）。线性I/O处理的情况下，容易造成系统瓶颈，只要看这个指标就可以了。例如，iSCSI或者NAS的外部存储器按照1 Gbit/s连接的话，按照Byte来换算，就是125 MB/s，这就是有可能形成系统瓶颈的物理带宽的上限值。

IOPS就是1秒钟处理I/O的量。在随机I/O处理的情况下，设备本身成为系统瓶颈的可能性更大，所以参照这一数值即可。

通常采用以上两种指标就可以进行基准测试。还需要注意的是，在查找性能瓶颈时，也需要把应答延迟率考虑在内。根据处理量不同，毫秒级性能提升也可能要考虑应答延迟率，即I/O处理从请求发出、结束处理到返回花费的时间。

用fio进行基准测试

fio[④]是对硬盘系统做基准测试的强有力工具。它的功能设定多而复杂，看上去好像不好用。以下介绍几种简单的功能。Windows系统常用的I/O基准测试工具是CrystalDiskMark。使用CrystalDiskMark进行批处理的方法，可以参考WinKey的网页[⑤]，以下我们就结合这个网页的内容做说明。

④ http://freecode.com/projects/fio
⑤ http://www.winkey.jp/

◆ fio的编译和运行

如图4所示，先要安装编译所需要的文件，然后下载fio的模块，编译以后再安装。

fio的应用主要有两种方式。一种是以命令形式不断输入设置内容，另一种是以fio命令调用制作好的任务文件，可以同时进行多种测试。以下着重介绍一下第二种方式。这种方式需要先把任务文件从WinKey的网页下载下来，然后用fio命令运行。

使用--output参数可以将输出文件定位到一定路径下，以便参考。另外，由于需要进行多种测试，因此以文件形式输出还是有一定好处的。

◆ 任务文件的重要设置

运行任务文件之前，先介绍一些重要的设置。这也是运行其他I/O基准测试工具前需要注意的内容。请参考下载后的任务文件（图5）。

size=1 g是基准测试使用的数据大小。RAID卡和外部存储器往往带有自己的缓存。1 g（1 GB）大小基本占用了所有的缓存，因此可能无法测出硬盘本身的性能数据。例如，缓存为1 G的情况下，需要测试使用全缓存时就应设置为512 m（512 MB），缓存不够用的情况下设置为10 g（10 GB）。以下，我们就将缓存设置为10 g来测试。

direct=1代表直接操作硬盘的模式。硬盘I/O使用的是操作系统上的缓存，将这一缓存未使用的部分给基准测试使用。测试硬盘性能的时候，需要设置为1。

◆ fio测试结果

测试结果被保存在指定目录下。笔者的执行结果如图6所示。

▼ 图4　fio的编译和运行

```
$ sudo yum -y install libaio-devel make gcc    ←下载软件包
$ curl http://brick.kernel.dk/snaps/fio-2.0.15.tar.bz2 | tar jx    ←下载工具
$ cd fio-2.0.15
$ make            ⎫←编译
$ sudo make install ⎭                                  ↓下载任务文件
$ curl -o crystaldiskmark.fio http://www.winkey.jp/downloads/visit.php/fio-crystaldiskmark
$ fio --output=./fio_result.txt ./crystaldiskmark.fio    ←运行基准测试程序
```

▼ 图5　任务文件

```
[global]
ioengine=libaio
iodepth=1
size=1g
direct=1
runtime=60
directory=/tmp/
filename=fio-diskmark
~以下为测试项目~
```

传输量(带宽）主要看Seq-Read和Seq-Write这两个结果。bw值就是线性读取时的传输速率，为381536KB/s。

IOPS主要看Rand-Write-4K-QD32和Rand-Write-4K-QD32的结果。IOPS值是随机写的操作结果，为2452IOPS。

本次使用1GB缓存、4个15000 rpm的SAS硬盘，并且以RAID10形式设置，测试结果是非常高速的。

fio具有很多可以设置的项目，可以通过修改任务文件来尝试。

内存的基准测试

能否只参考内存容量

确定服务器等级时，需要先选择内存容量。之所以很少测试服务器的内存速度，是因为内存处理速度是很快的，相比之下，其他地方更容易形成系统瓶颈。因此内存的处理速度对应用程序的影响不大。

内存处理速度主要受主频的影响。例如，PC3-12800主频为1600 MHz。主频越大应该越快。还需考虑使用通道问题，也就是内存插在了哪个插槽，由几片组成。另外，内存处理速度与服务器的年代也有关，购买前最好参考产品说明书。比较内存的基准测试结果，正确地使用内存通道，这点更加重要。

用SysBench进行基准测试

SysBench⑥不单对内存，对CPU和其他方面（数据库OLTP）都可以进行测试。

◆ SysBench的编译与运行

如图7所示安装编译所需文件。下载SysBench源程序，编译后安装好。由于在这里不进行OLTP的测试，因此在配置时应设置为--without-mysql。如果需要测试数据库，那么在安装MySQL程序包后把--without-mysql删掉就可以了。

内存I/O的测试与硬盘I/O一样，由读处理、

⑥ http://SysBench.sourceforge.net/

▼ 图6　fio的运行结果

```
~省略~
Seq-Read: (groupid=0, jobs=1): err= 0: pid=4286: Fri May 17 04:34:23 2013
  read : io=10240MB, bw=381536KB/s, iops=372 , runt= 27483msec
    slat (usec): min=58 , max=458 , avg=61.34, stdev= 6.55
    clat (usec): min=165 , max=218217 , avg=2621.01, stdev=4375.96
     lat (usec): min=231 , max=218287 , avg=2682.53, stdev=4376.68
~省略~
Rand-Write-4K-QD32: (groupid=7, jobs=1): err= 0: pid=4296: Fri May 17 04:34:23 2013
  write: io=588888KB, bw=9811.3KB/s, iops=2452 , runt= 60022msec
    slat (usec): min=4 , max=17111 , avg= 9.92, stdev=103.52
    clat (usec): min=1 , max=40593 , avg=13034.62, stdev=6690.25
     lat (usec): min=24 , max=40601 , avg=13044.69, stdev=6689.90
~省略~
```

写处理、线性I/O处理、随机I/O处理四种情况组成。内存I/O基本上是随机完成的，所以不会太在意使用的是线性处理还是随机处理。

内存I/O测试运行选项是--test=memory。设置读处理或者写处理只要把--memory-oper设为Read或者Write就可以了。把--memory-access-mode设置为seq（线性I/O处理）或者rnd（随机I/O处理）。

◆ SysBench的测试结果

测试结果显示在标准输出装置中。笔者执行了随机写处理（rnd/Write），结果如图8所示。传输量 的最终结果是3591.55 MB/s。

这里的内存使用了4通道中的2通道，并非最佳配置。主频为1333 MHz。

SysBench不仅适用于内存，还适用于CPU等。不妨试一试。

▼ 图7　SysBench的编译和运行

```
$ sudo yum -y install libtool make gcc    ←下载软件包
$ curl -L http://downloads.sourceforge.net/project/sysbench/sysbench/0.4.12/sysbench-0.4.12.⏎
tar.gz | tar zx   ←下载工具
$ cd sysbench-0.4.12                ┐
$ libtoolize --force --copy         │
$ ./autogen.sh                      │
$ ./configure --without-mysql       ├←编译
$ make                              │
$ sudo make install                 ┘
                                                                          运行基准测试程序
$ sysbench --test=memory --num-threads=1 --memory-oper=read --memory-access-mode=seq run ←┘
$ sysbench --test=memory --num-threads=1 --memory-oper=write --memory-access-mode=seq run
$ sysbench --test=memory --num-threads=1 --memory-oper=read --memory-access-mode=rnd run
$ sysbench --test=memory --num-threads=1 --memory-oper=write --memory-access-mode=rnd run
```

▼ 图8　SysBench的运行结果

```
Operations performed: 104857600 (3677744.60 ops/sec)

102400.00 MB transferred (3591.55 MB/sec)  ←运行结果

Test execution summary:
    total time:                          28.5114s
    total number of events:              104857600
    total time taken by event execution: 20.8725
    per-request statistics:
         min:                                  0.00ms
         avg:                                  0.00ms
         max:                                  0.30ms
         approx.  95 percentile:               0.00ms

Threads fairness:
    events (avg/stddev):           104857600.0000/0.00
    execution time (avg/stddev):   20.8725/0.00
```

网络基准测试

从服务器到网络，最初关联的是网络接口NIC（Network Interface Card）。

网络基准测试的必要性

服务器的网络接口NIC（Network Interface Card）一般都是1000BASE-T，即1 Gbit/s。面向服务器的网卡基本都是这个水平，因此是没有必要对网卡本身做基准测试的。另外，伴随闪存等外设的高速化和低价格化，10 Gbit/s的产品会不断出现。

这里对10 Gbit/s的网卡做一次基准测试。所用方法除了测试网卡外，也可以对整个网络做基准测试。例如，如果有L2交换集线器和L3交换集线器构成的局域网，可以对它的互联网功能进行测试。

用netperf进行基准测试

netperf[7]是一款功能非常强大的基准测试工具。设置功能繁多，看上去不太好用。但是，如果只测试传输量（带宽），只通过缺省设置就可以完成。

◆ 网络结构

netperf是对应客户端和服务器的网络工具，也就是说，至少需要两台机器才能做基准测试。这次测试的网络结构参考图9，使用DAC把两台服务器连接，每台都装有Intel 10 Gbit/s的网卡X520-DA2。因为是SFP+的10 Gbit/s网卡才使用DAC连接，所以如果是1000BASE-T或10 GBASE-T，则可以直接使用非交叉型LAN网线（最近的新型网卡支持直通型LAN网线进行连接）。

当然，两个机器之间加一个交换集线器也未尝不可。

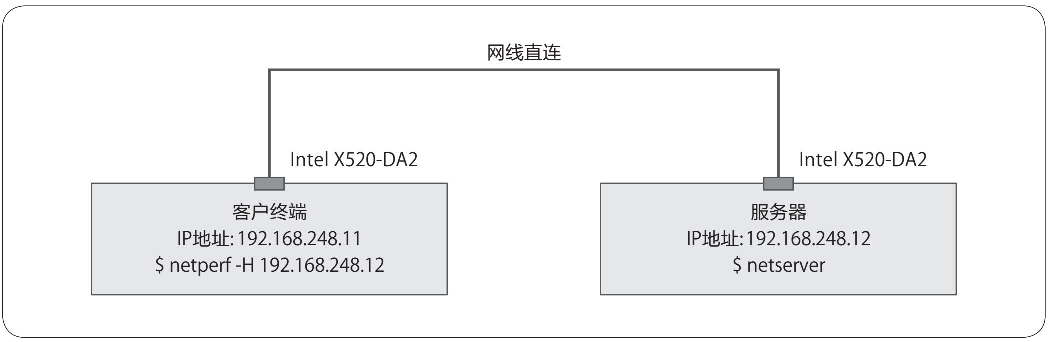
◆ netperf编译与运行

图10是安装编译所需的软件包。然后下载netperf源程序，编译后再进行安装。这两台机器都需要安装这个软件。

然后，在服务器上运行netserver命令。缺省端口号为12865，如果使用了iptable等网络过滤器（防火墙），那么请将那个端口打开，或者先暂停网络过滤器程序。

⑦ http：//www.Netperf.org/netperf/

▼ 图9 网络结构图



服务器准备好以后，就可以开始在客户端进行基准测试了。以10秒间隔的TCP通信为缺省设置。

◆ netperf的测试结果

测试结果显示到标准输出装置。

笔者的运行结果如图11所示。最终结果参考Throughput项目，单位是10的6次方bit/s，即Mbit/s。以10 Gbit/s连接，传输率基本接近物理的临界值。10 Gbit/s果然是10 Gbit/s的传输率啊。

netperf还有许多其他功能，值得以后尝试。

结束语

以上介绍了各主要组件的基准测试。测试完成之后最重要的是确认结果的正确性。至于为什么比预想值偏高或者偏低，需要进行细致的分析。要明确系统瓶颈在哪里，缓存是否生效。这些经验会加深对服务器结构的理解，从而构建一个安定的运行环境。

▼ 图10　netperf的编译与运行

```
・在服务器与客户端两边运行
$ sudo yum -y install make gcc   ←下载软件包
$ curl ftp://ftp.netperf.org/netperf/netperf-2.6.0.tar.bz2 | tar jx   ←下载工具
$ cd netperf-2.6.0
$ ./configure      ┐
$ make             ├←编译
$ sudo make install┘

・在服务器上运行
$ netserver

・在客户端上运行
$ netperf -H <服务器IP地址>

・在服务器上的运行结束
$ killall netserver
```

▼ 图11　netperf的结果截屏

```
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.248.12 () port 0 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.00    9410.82
```

分布式数据库“未来工房”

第1回 只使用Riak和Nginx搭建的静态文件服务器——其构造与性能测试结果

草庵辞旧主
从此居中多倩影

文/上西康太（Basho日本株式会社　kota@basho.com）　译/苏祎

在搭建网站时，如果更新不多会用静态文件服务器，但如果要扩容，处理巨大的流量是件比较复杂的事情。负载均衡、磁盘冗余、容量规划、横向扩展、缩减，工程师需要考虑的问题非常多。在这里，我们将使用基于GitHub pages中表现出色的Riak，在不编程的情况下，搭建既可以维持稳定延迟，又可快速扩展的静态文件服务器网站。

概要

大家一定都还记得初次搭建Web服务器的时候，先安装Apache httpd，将写入了hello的index.html放到指定文件夹，然后在浏览器里打开它。或者也有不少读者可能记得另一种方法，即用FTP客户端将HTML文件上传到Geocities。

静态文件服务器是创建Web网站的基础。看上去非常简单，但其实相当深奥。比如说，很多人可能有过这种经历，等HTTP下载Linux的ISO镜像等了一个晚上，最终通过tarball①下载的时候却发生硬盘故障，或因为超负荷服务器宕机了。实际上，Web网站的服务器需要使用到负载均衡、磁盘冗余、容量规划、横向扩展、网络消耗等众多技术。

其实这些问题，除去节省网络消耗外，只要使用两个软件就可以解决了。那就是Nginx和Riak。通过Nginx完成负载均衡，而磁盘冗余、容量规划、横向扩展，都可以通过Riak实现。Riak容易被认为是单纯的KVS（Key-value Store），并被拿来和RDB比较，但其接口为HTTP，并且可将数据以记录为粒度进行冗余化，只要增加机器就可以实现横向扩展。既可以作为数据库处理较大数据的记录，也可以当作存储几千到几百万级别小数据的对象数据库来使用。

本连载会介绍一种方法，使用基于GitHub pages中表现出色的Riak，在不编程的情况下，搭建既维持稳定延迟，又快速扩展的静态文件服务器网站。

网站设计

我们假设要搭建静态文件服务器的网站的域名为foo.example.com。想要在这个网站中实现的是：

1. 存储大量静态文件。
2. 通过foo.example.com/path/to/the/file这样的URL定位每一个静态文件。

① 使用tar命令打包的文件。工程师俚语。

3. 静态文件的大小最多也不超过几MB，平均为数百KB。

4. 静态文件的个数在1000～1 000 000 000之间，起初先小规模启动。

5. 所有图片都会被以相同的概率随机被读取。

6. 不会特意地去删除图片（简单起见）。

7. 响应时间的要求为，99%的请求的响应时间控制在500 ms以内。

8. 争取无服务停止时间。

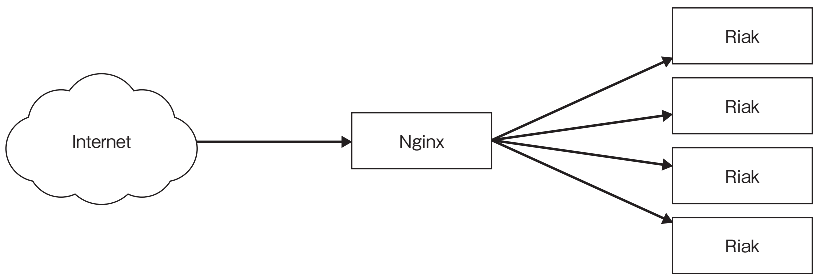
以上内容，特别是第4条的横向扩展和第8条的停止时间，如果采用普通的文件服务器及Apache的HA（High Availability）架构，实现起来会比较困难。即便导入了Nginx这样的负载均衡器，需要自行设计及操作数据库分片，而第8条的不停服务维护，以及磁盘故障冗余的建立和负载均衡的分发设定也都很复杂。所以，在这样的分工前提下，依靠功能本身更为丰富的Riak即可以简化系统整体。

Riak是基于2006年由Amazon公开的Dynamo设计出来的。它的主要特性为，通过Consistent Hashing进行水平分布、扩容，通过复制实现冗余，通过稳定的响应提供高可用性等。除此之外，它通过使用JavaScript执行MapReduce，具有与除Join之外的SQL同等的表现能力。

Riak具有HTTP接口，在Riak中存储的数据可以通过浏览器原样展示出来。因此，如果这样就在互联网上公开的话，不仅通过HTTP的GET方法可以访问Riak，PUT和DELETE方法也可以不经过认证执行了。为了防范任意用户对数据的删除、更新，首先，使用Riak，有必要通过NAT（Network Address Translation）来构建一个确保不能直接通过互联网访问Riak的网络。

因为大部分的负载均衡器都能设置为只处理特定的HTTP方法，拒绝其他方法，如果读者自身比较熟悉的话，使用那个也没问题。这里，我们使用非常流行且高速的Nginx来作为例子。其实在Nginx之外，还有很多的免费/商业的负载均衡器，例如HAProxy、Varnish、LVS和

▼ 图1　foo.example.com的架构例子



小贴士

在使用云服务的情况下，因为环境的不同，使用方式也略有不同。如果是AWS的EC2，使用Marketplace的AMI比较好。在EC2环境下，所有的节点都分配了一个全球IP地址，最好使用一开始就已经利用IPSec限制访问的官方AMI。若选择IDC Frontier，最好使用cloud selftype官方API或VPC（Virtual Private Cloud）。使用NTT通信的Cloud^{n}或Nifty云等以CloudStack为基础的平台的话，可以让虚拟机一开始就被隔离到了私有网络中。

BigIP等，读者只要使用自己常用的就行了。

把Nginx设置为只允许GET方法，PUT、DELETE等其他方法都拒绝。对于实际的文件更新，这里为简单起见，假设不通过负载均衡器，而是使用curl直接访问Riak集群来进行维护。

理论上来说，除了后面介绍的Riak集群，根据使用场景的不同，下面这些内容也是会用到的：

- 用来上传文件的页面（包含认证及上传的表单）。
- 监视Riak集群的系统（Zabbix等）。
- 针对系统扩展的部署系统（Puppet、Chef等）。
- 获取、管理访问日志以及切断非法访问。
- 为了降低负载使用的CDN（Contents Delivery Network）。

验证操作的软件环境如下所示：

- Ubuntu Linux LTS 12.04/x86_64
- Riak 1.3.1
- Nginx - 1.1.19（Ubuntu的apt版本）
- curl

大部分硬件都能运行上面的软件需要的环境。并且，Riak几乎支持所有的Unix系平台。运行Riak没有什么特殊的最低硬件条件，比如：

- 8核的x86 CPU
- 内存16～32 GB
- 硬盘8 TB
- 1 GB的网卡

或

- 32核的CPU
- 内存64～128 GB
- 硬盘72 TB
- 10 GB的网卡

上述配置都是可以的。磁盘方面，需要使用Raid0配套LVM、ZFS这样的目录管理系统。并且，这次使用Bitcask作为后端，如果静态文件数量太多的话，内存的消耗量在纸上计算出来会比较好[2]。说到磁盘大小，可以通过需部署文件的大小乘以个数来预估：

每台的磁盘大小 ≥ 文件的平均大小 × 每台的文件数 × 3 + 余量（20%）

这样估算是可以。（此数值受故障率及网络组成影响，不能一概而论。想必读者都知道运行的稳定性及运行成本之间是相互制约的。）事实上，这需要综合考虑硬件的MTBF（Mean Time Between Failures）和MTR（Mean Time to Recovery），以及网络带宽来进行总体设计，以便发挥最高的性能。

Riak的准备与启动

下面介绍实际搭建网站的步骤。Riak运行所需的最低环境[3]要求为64位CPU、4GB RAM、多个HDD、高速网络。虽然Basho推荐Riak台数最少为5台，但这边为了便于验证操作，只以3台（10.0.1.11，10.0.1.12，10.0.1.13）作为例子。并且准备好一台Nginx的机器（10.0.1.10）。此处读者可依据自身的环境替换为其他配置。

截至2013年4月Riak的最新版本号为1.3.1。如果是Ubuntu Linux 12.04或者APT环境，可以使用Basho提供的apt-line来进行安装[4]。

② http://docs.basho.com/riak/latest/references/appendices/Bitcask-Capacity-Planning/
③ http://docs.basho.com/riak/1.3.1/tutorials/System-Planning/#Hardware
④ http://docs.basho/riak/1.2.0/tutorials/installation/Installing-on-Debian-and-Ubuntu/

```
$ sudo curl http://apt.basho.com/gpg/ 
basho.apt.key | sudo apt-key add -
$ sudo bash -c "echo deb http://apt. 
basho.com $ (lsb_release -sc) main > 
/etc/apt/sources.list.d/basho.list"
$ sudo apt-get update
$ sudo apt-get install riak
```

这样，Riak就安装好了。如果在不能访问网络的环境下使用的话，可以从Basho的网站[⑤]下载，并使用dpkg命令进行安装。

```
# sudo dpkg -i riak_1.3.1-1.amd64.deb
```

这样，这台服务器上就装好了。

Riak在大多环境下都不需要优化参数（这里所指的是为了获得最优参数不断地进行性能测试，但在Linux环境下为了获得最佳性能，需要变更多个参数）。详细请参见参数的设定[⑥]。

```
# sudo dpkg -i riak_1.3.1-1.amd64.deb
```

接下来，为了增大进程能打开的最大FD数量，需要对`/etc/security/limits.conf`进行编辑。

```
riak    hard    nprc    100000
riak    soft    nprc    100000
```

另外，因为会影响硬盘的I/O性能，内核的I/O调度器需要设置为deadline。`sda`部分，使用的是`/etc/riak/app.config`中设定的数据目录所在的设备。

```
# sudo echo deadline > /sys/block/sda/ 
queue/scheduler
$ cat /sys/blok/sda/queue/scheduler
```

根据环境需要对`/etc/riak/app.config`与`/etc/riak/vm.args`进行设定。需要设定为接受HTTP请求要绑定的IP地址和Erlang通信库需要使用到的IP地址（默认情况下会已设置为127.0.0.1的本地回环地址，从外部无法使用）。

app.config的第33行（riak_core区块的http项目）可以设置HTTP的IP，将其设置为自己Riak服务器的IP地址。

```
{http, [ {"10.0.1.11", 8098} ]},
```

另外，数据目录原来应是`/var/lib/riak`，如果已经定下数据用的分区路径，就要调整这个地方。是在默认bitcask区块的data_root项目外。

接下来，设置`/etc/riak/vm.args`。

```
## Name of the riak node
-name riak@10.0.1.11
+zdbbl 131072
```

这样做是为了在Erlang的通信库里指明IP地址。第二个`+zdbbl`[⑦]是Erlang/OPT的通信库的发送队列的大小（KB单位），默认是1024 KB，最大可设置为2 GB。

通过这些步骤，设置已基本完成，接下来在所有服务器上启动Riak进程。

```
[10.0.1.11]
$ sudo riak start

[10.0.1.12]
$ sudo riak start

[10.0.1.13]
$ sudo riak start
```

这样3台服务器上就完成了启动，但其相互识别为不同的Riak集群，所以需要使用`cluster`命令识别为同一集群。从思路上来说，在图1中显示的一台服务器10.0.1.11的集群与另两台汇合成了一个集群。

⑤ http://docs.basho.com/riak/latest/downloads/。
⑥ http://docs.basho.com/riak/latest/cookbooks/Linux-Performance-Tuning/。
⑦ http://docs.basho.com/riak/latest/cookbooks/faqs/logs-faq/#riak-logs-have-busy_dist_port-messages。

```
[10.0.1.12]
$ riak-admin cluster join riak@10.0.1.11

[10.0.1.13]
$ riak-admin cluster join riak@10.0.1.11

[10.0.1.11]
$ riak-admin cluster plan
$ riak-admin cluster commit
```

cluster plan命令是用来规划vnode数据管理单位的配置命令。没有任何数据的状态下，最初构建集群时，并不会感觉到有太大的必要性，但在积累到一定数据无法停止系统时，用来控制再配置时就很有用了。执行cluster commit的话，就会实施plan里计划的再构建。具体来说，随着时间的转移，会实现vnode的转移。通过执行以下命令：

```
$ riak-admin transfers
```

这样就可以观察到转移的具体情况。如果希望调整再配置的速度，就通过执行如下命令：

```
$ riak-admin transfer-limit
$ riak-admin transfer-limit 1
```

即可确认和调整再配置的带宽。transfer-limit的默认值为2，这是可以同时连接的数量（等于可同时移动的vnode），所以会成为整体的再配置所需要的时间、网络带宽以及机器的负荷之间相互制约的结果。在再配置的过程中这个值也是可以变更的，所以可以边观察transfers的情况及服务器的负荷边进行值的调整。

这样，服务器就可以启动并且顺利作为集群运行了。确认的方法有多种。首先，使用Riak的运用命令的方法如下：

```
$ curl -X PUT http://10.0.1.11:8098/ ⏎
buckets/spam/keys/ham -d 'egg'
$ curl http://10.0.1.12:8098/buckets/ ⏎
spam/keys/ham
egg
$ curl -X DELETE http://10.0.1.13:8098/ ⏎
buckets/spam/keys/ham
$ curl http://10.0.1.11:8098/buckets/ ⏎
spam/keys/ham
not found
```

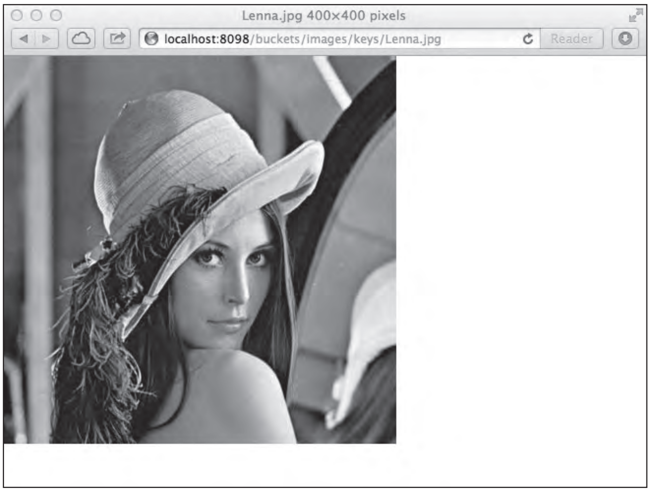
对Bucket名为spam、key名为ham的记录，可对到3台机器的访问进行确认。

初试文件的上传

首先，在不使用Nginx的情况下，上传样本图片（图2）。

```
$ curl -X PUT http://10.0.1.11:8098/ ⏎
buckets/static/keys/Lenna.jpg \
  -H "Content-type: image/jpeg" ⏎
  --data-binary @Lenna.jpg
$ gnome-open http://10.0.1.11:8098/ ⏎
buckets/static/keys/Lenna.jpg
```

▼ 图2　因为Riak有http接口，所以可以使用浏览器确认数据



如果使用非gnome窗口管理器，或其他OS，可以用浏览器打开这个URL，应该就能确认上传的图片了。

Nginx的设置

Riak的基本功能的可用性已经得到确认，接下来进入到Nginx的设置阶段。设置Nginx需要完成以下两点：

1. 将GET的请求平均转发给Riak集群。
2. 拒绝PUT、POST等和更新相关的请求。

首先在10.0.1.10的机器上安装Nginx。

```
[10.0.1.10]
$ sudo apt-get install nginx
```

为了设置上面的内容，编辑`/etc/nginx/site-enabled/default`。

首先将HttpUpstreamModule[8]的设定记述为`upstream backend`，即可指定分配HTTP请求的Riak服务器。

```
upstream backend {
      server 10.0.1.11:8098;
      server 10.0.1.12:8098;
      server 10.0.1.13:8098;
      keepalive 4096;
}
```

如上，将IP地址和端口号罗列就可以了。另外，`keepalive`[9]的作用是，以HTTP KeepAlive保持4096条TCP连接。

接下来，将GET以外的请求，都设定为不可通过，通过使用`limit_except`[10]记述到`server`区块。若网站的FQDN是foo.example.com，那就这样配置：

```
server {
   listen   80;
   server_name foo.example.com;

   location / {
       limit_except GET {
            deny   all;
       }
       proxy_pass http://backend;
   }
}
```

上述的意思是“监听80端口”，这个服务器的FQDN是foo.example，接受方法只有GET，向`proxy_pass`指定的服务器通过负载均衡进行传输。

这样，就可以访问http://foo.example.com/buckets/static/keys/Lenna.jpg所指定的图像了。实际的应用程序中，会更宽松地进行路径指定。如果需要在Riak的Key或Bucket中包含“/”字符，可以在Nginx里重写URL，把编码后的文字作为bucket或Key的名字就可以了。

使用basho_bench进行压力测试，增加服务器进行扩展

接下来对搭建的静态服务器性能进行实际评估。评估HTTP服务器性能，使用ab（apache bench）比较方便，这里为了减少磁盘缓存的影响，最好对图片进行随机访问。所以，这次主要介绍Basho为评估Riak性能所开发的basho_bench。

准备

首先，向Riak上传100万张图片作为静态文件样本。

[8] http://wiki.nginx.org/HttpUpstreamModule
[9] http://nginx.org/en/docs/http/ngx_http_upstream_module.html#keepalive
[10] http://wiki.nginx.org/HttpCoreModule#limit_except

▼ 图3　用不同名字上传1,048,576个相同图片的ruby脚本

```
n = 1024 * 1024
(0..n).each do |i|
  url = "http://10.0.1.11:8098/buckets/static/keys/#{i}.jpg"
  print "#{i}/#{n} th image to #{url}\n"
  print `curl -X PUT #{url} -H "content-type: image/jpeg" --data-binary @lenna.jpg`
end
```

实际上不必使用完全不同的图片，文件名使用自动命名就行了。这里为了方便basho_bench使用，在Riak上使用的是连续数字的文件名1.jpg，……，1048576.jpg。图片的大小平均为25 KB，复制2份，总计75GB的文件被上传到集群中。

我们准备了一个简单的脚本来上传文件。其实什么语言都可以，这里挑选了Ruby（见图3）。

同时，开始basho_bench的准备工作。我们准备了另一台服务器来承担负载，并安装好Git和Erlang。

```
$ git clone git://github.com/basho/ ↩
basho_bench.git
$ cd basho_bench
$ make
```

这样就生成好了名为basho_bench的可执行文件。basho_bench的配置文件如下：

```
{mode, max}.
{duration, 30}.
{concurrent, 64}.
{driver, basho_bench_driver_http_raw}.

{key_generator, {int_to_str, {uniform_int,
1048576}}}.

{http_raw_ips, ["10.0.1.10"]}.
{http_raw_prot, 80}.
{http_raw_path, "/buckets/static/keys"}.
{http_raw_params, ".jpg"}.

{operations, [{get, 1}]}.
```

这里不对每个项目都详细说明，只要理解以下内容即可，对安装了Nginx的服务器，使用 http://10.0.1.10/buckets/static/keys/{i}.jpg（i是随机从1～1 048 576中挑选出来的）地址以64的并发，连续访问30分钟。

还有，为了降低读取负载，把读取访问数设置为1。若读取和写入经常发生冲突也可以设定为超过半数。但我们这边考虑的访问是Write once、Read many，所以没有必要这样做。

```
$ curl -X PUT http://10.0.1.11:8098/ ↩
buckets/static/props \
  -H "content-type: application/json" ↩
-d '{"props":{"r":1}}'
```

这样，就在名为static的bucket里设置了r=1。

实际使用的硬件环境，为如下配置的虚拟机：

- Intel Xeon CPU，2.40 GHz，4核
- 内存8 GB
- 硬盘数据区域128 GB

实验

执行basho_bench的基准测试，只要执行如下脚本：

```
$ ./basho_bench httpraw.config
```

如果要马上在命令行上显示图像，就执行：

```
$ ./priv/gp_throughput.sh -t dumb
$ ./priv/gp_latencies.sh -t dumb
```

这会在终端上用文本绘制出图像。这个命令在没有Xforward的服务器上很有用。

这次评估，通过在2到8台Riak之间进行变化，来测定相应的吞吐量和延迟。可以想象的是，随着Riak运行台数的增加，其负荷也能更加分散，吞吐会线性上升，但可以限制在一定数值下。同时，在实际超过6台左右时，每一台的数据量将低于物理存储量，所以性能会更好。

结果

首先，从吞吐量来看，机器数量在两台时是700 qps，8台时是2700 qps，表明随着机器台数的增加，吞吐量可以稳定地进行扩展（见图4）。

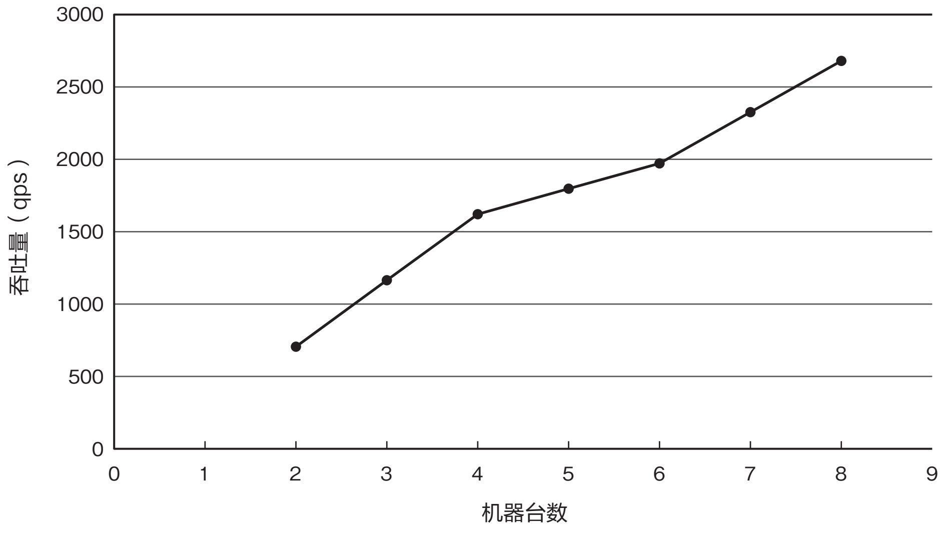
事实上，因为lenna.png的大小约为25 KB，所以basho_bench通过Nginx，获得了67.5 MB/s的数据。

因为是静态文件，处理这样规模的流量还是使用CDN（Contents Delivery Network）比较稳妥。

图5也表明了延迟的状况。不同台数的机器承担相同负荷，可以看出台数少的时候，最长延迟情况很差。8台的时候，99%的请求在400 ms以内就返回了。或者说，随着台数的增加，延迟和吞吐量都有提升，可以认为负载均衡的目的已经达成。

另外，在实验进行过程中，我费心于调整Nginx，设置KeepAlive，设置OS，并多次往集群中增加Riak服务器，结果测试到6台时发现错误，再次从两台开始测试。这样的操作反反复复进行了很多次。但是就算这样，100万个文件没有丢失过，可以顺畅地进行增加、删除服务器的操作。或者说，2、4、6、8台测试的过程中，Riak整个系统没有一次停止过。这样看来，相信大家也能明白不间断地运行Riak是多么地容易了。

▼ 图4　台数变化过程中的HTTP GET的吞吐量



GitHub pages

本文只介绍了最简单的方法。关于实际的服务Github pages是如何运作的，在Erlang Factory 2012上Github的工程师Jesse Newland的演讲中有详细说明 。实际上，Github pages的静态内容都缓存在名为Fastly的CDN上（图6）。并且，因为有些路径包含"/"字符，可以使用Webmachine架设一个简单的HTTP服务器，把路径和Riak的数据一一对应变换，对静态内容进行简单的版本管理。

使用了Riak，GitHub Pages现在可以维持着快速响应的同时进行扩展。

其他途径

Nginx上也有标准的静态文件传送功能，利用这项功能，会十分方便。使用RAID提高磁盘的可靠性，再加上一定的缓存功能，也可以进行高速的静态文件传送。同样，把NFS上的数据mount到多台机器上，并用Apache或Nginx做文件服务器也是可以的。

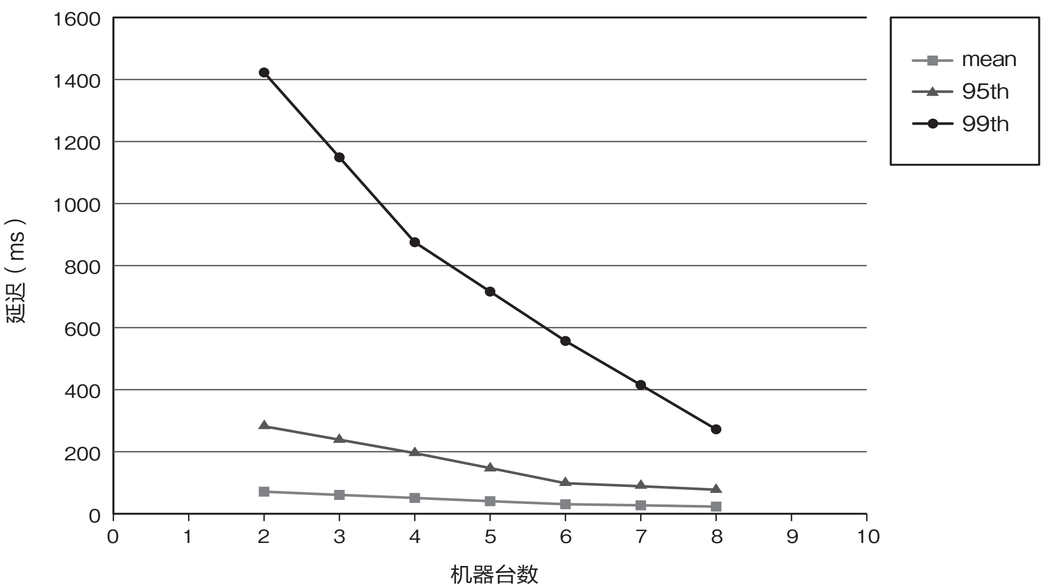
或者，也可以使用3月公开的OSS版Riak CS。[11]它是和Amazon S3兼容的分布式存储方案，由于后端使用了Riak，所以具有了与Riak一样的容错性，并可使用S3相关的工具。它所支持的最大对象可以达到5TB。

总结

本文介绍了使用Riak和Nginx搭建支持横向扩展静态文件服务器的方法。因为使用了Riak，数量和负载两方面都能实现扩展和分

⑪ http://basho.co.jp/products/riakcs

▼ 图5　台数变化过程中的HTTP GET的延迟



▼ 图6　Github Pages使用Fastly作为CDN

```
$ dig basho.github.io
…省略…
;; QUESTION SECTION:

;basho.github.io.          IN   A

;; ANSWER SECTION:

basho.githum.io.     3600   IN   CNAME    github.map.fastly.net.

github.map.fastly.net.      30   IN       A       103.245.222.133
```

流。并且，可以利用Riak保持稳定延迟，利用Nginx的简单配置实现负载均衡。

虽然这次只使用了8台服务器，但还是有一些商业应用使用了50到100台服务器，甚至扩展到更多。请一定挑战一下更大的集群。还有，这次数据量只固定到了25GB，随着服务器台数的增加，总的数据量也可以相应增加。可以根据系统需求调整硬盘大小或服务器台数。

Erlong/OTP和Riak由于自身设计方面的原因，读取性能和吞吐并不太高。但在实际使用中，比起使用最耗资源的高性能数据库和内存数据库，选择CDN服务商提供的静态文件发布服务，会即高效又廉价。高峰时的读取请求委托给CDN，自己保存原始数据，应付缓存未命中时所使用的数据存储Riak，应该也相当划算。

虽然现在不能像以前那样，个人有机会运营文本、图片的网站，但当突然有数百万个文件出现在眼前时，本文描述的基本思考方法能对读者有所帮助的话，那就再好不过了。

参考文献

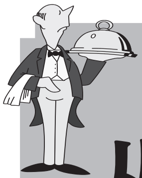
[1] 官网是http://www.nginx.org

[2] 参见http://github.com/basho/riak

[3] 参见http://pages.github.com

[4] Giuseppe DeCandia等. Dynamo: Amazon’s highly available key-value store. In SOSP’ 07

[5] Jesse Newland, Rewriting GitHub Pages with Riak Core, Riak KV, and Webmachine. In Erlang Factory SF2012.（http://www.erlang-factory.com/conference/SFBay2012/speakers/JesseNewland）



按部就班真的可行吗？

从小规模工程学习活用 Jenkins



第1回 真的有必要用程序来做这些吗？

你听说过 Jenkins 吗？可能有人听说过它的前身 Hudson。包括 Jenkins 的主页（Jenkins-ci.org）在内，网上有很多有关 Jenkins 的介绍，但是仅靠参照这些网站，想在自己的环境中方便地使用 Jenkins，多少还会有些问题。如果仅靠程序的帮助，就能让每天的工作变得简单，是多么美好啊！但是事实往往没有这么简单。

文/岛崎聪（株）XVI　Twitter @sato_c　译/阿逸

小规模工程的烦恼

说起使用 Jenkins 的开发环境，很容易让人联想到大规模的工程。杂志和网站上介绍的 Jenkins 的使用方法，大部分是由多名程序员分模块开发软件，并向 Subversion 或 Git 提交代码后，将单体测试和结合测试交给 Jenkins。但笔者现在使用的环境却恰恰相反：我们并不需要很多的开发人员，以最小限度编写程序，将更多的时间用在制作、整理数据上。

这里的数据是指影像或图像、音乐或效果音以及消息中的文字，以及影像文件的播放方式。数据的种类和数量都很多，一个人做的话恐怕非常困难，并且会耗费大量的时间。简单重复的操作越多，耗费的时间越长，不小心犯错的可能性也随之提高，因此就需要更多的时间来检查确认。面对如此繁琐的工作，你肯定希望找到一种代替作业的方法吧。

并且，数据的制作需要大量的重复性劳动。制作一次就能完成的数据，恐怕在测试中也几乎从来没有过吧。考虑到这样的状况，我们希望能够节省运行批处理文件或者从命令行输入 make 的时间。如果能够省略上述操作，实现完全自动化构筑制作数据的环境，那会是相当有用的。因为有完备的环境，接下来的数据制作工作就能交给电脑来完成。这样的方式，你认为怎么样呢？

笔者以程序员的身份参与项目的开发，同时兼任着该系统环境的构建和管理的工作（小规模的开发中，由程序员兼任此类工作也是很常见的），因此考虑使用 Jenkins 来实现上述机制。

这样的情况下，你会怎么做呢？

程序开发过程中需要少量数据时，你会怎么做呢？因为是很少量的数据，一般会由程序员用文本编辑器手动输入，不用文本编辑器的话就是用 Excel，一般不都是这样吗？其实，这是一个不好的开始，随着开发过程的深入，

会逐渐变成依靠手工制作数据。并且如果在部署到正式环境的过程中，还是这样做的话，很容易造成数据的不整合，引起各种问题。制作数据的程序员如果不能随机应变，提高制作数据的效率，那么也会非常辛苦。

你可能会想到“用批处理文件来处理就好了”或者“用脚本实现自动运行就好了”。的确，开发人员最初会认为自己拥有一个完美的开发环境，事实上，这很难实现。笔者之前的工作中就遇到过这样的情况：完全不需要手动干预的批处理文件中，用call的形式调用了其他的批处理文件，这些被调用的批处理，将计算结果写入到环境变量中来传递。结果就像函数调用一样地使用结构化的批处理文件。如果顺利，这样的方法会很有效。一旦出现了问题，会导致除了批处理文件制作者之外，别人很难理解，之后的数据制作和批处理文件的维护都非常麻烦。

因此，批处理文件逐渐被易于理解的脚本所取代，其实如果可以一开始就用脚本语言来编写的话就更好了。并且脚本的运行也更加快速，能很大程度地改善开发环境。也可以通过修改数据结构和式样等，来使数据维护工作变得简单。这样在测试结束后，部署到正式的运行环境时也会更加方便。

共同开发中的差等思考

笔者常听到身边的设计师说：“我只是画画的，之后如何使用图片我不知道，你来弄一下。”的确，从头开始学习图片数据的加工处理确实非常困难。对方没有理解时，需要反复地对处理步骤从头进行说明，这对双方来说都是一种负担。说明的一方会觉得这是在浪费大家的宝贵时间，逐渐会觉得这种事情与自己无关，于是事不关己高高挂起。诸如此类的烦恼，在项目的后半程尤为突出。

并且管理方面也开始出现问题，例如：重要画面中遗漏了图片、图片或影像的版本陈旧、图片的编解码有问题等。诸如此类的问题在项目渐入佳境的同时开始逐渐显现。修复问题时，数据管理的负担会越来越重，变成了相当棘手的问题。

用Jenkins来解决吧！

本连载将介绍通过导入Jenkins来解决上面描述的种种困局，当然文中可能夹杂着笔者对自己所处状况的种种感想。

因为介绍的是根据笔者自身所处状况来使用Jenkins，所以可能和Jenkins入门之类的文章相去甚远，而且，Jenkins的使用方法本来就应该是多样化的。笔者要写的东西总结如下：提高小规模工程的开发效率，及Jenkins的相关使用方法。

在这样的环境下工作

笔者现在从事的是娱乐系统的开发，这套娱乐系统简单地说，就是根据用户的输入播放相应视频。笔者作为程序员参与此项目的开发，除笔者之外，还有设计师数名、企划人员、项目管理人员（图1），开发团队最多10人左右。影像、音乐和声音素材是向专门的公司购买后再进行编辑的，暂时不自行制作数据。

设计师的工作是设计画面和编辑音像资料。

▼ 图1　项目的职责分配



他们对于编辑音像和画像的环境、工具等相当了解，但在程序方面知之甚少。几年前，他们之中可能有人只知道如何启动、关闭电脑以及Photoshop的使用方法。最近，随着电脑的操作简易化，工作以外使用电脑的频率增加，只会开机关机的人已经很少了。但还是有很多人不知道如何将自己制作的音像数据转换成程序能够使用的数据，能够独自完成数据制作、转换的人很少（虽然不是完全没有）。

策划的主要工作是考虑产品的内容，在诸多限制下尽可能地将内容设计得有趣，这是胜负的关键。所以策划一般不会考虑细节问题，而会将注意力集中在自己的策划工作上。

项目管理者，顾名思义，主要工作就是把控项目的进度。他脑中考虑的是如何在规定期限内使开发顺利运转并完成项目。

开发就是将各种人集合到一起，将他们的工作关联起来。所以他们相互是否能够顺利地交流、协作，是需要注意的地方。

这些事都由程序员来做，岂不是负担很重?

身处这样的开发环境下，与设计师间交流沟通的程度与深度，需要事先和设计师进行说明并取得理解。策划会时不时地要求“想看一下现在产品的内容，要怎么做好呢”，这时需要将当前的产品、数据呈现给他看。同事的开发环境和自己在用的环境不尽相同时（事实上也不可能完全相同），必须设法解决，比如将那个人专用的程序或者Excel的宏打包发送过去等。

因此可以这么说，当设计师和策划摊开双手说“我不知道怎么处理”的时候，都必须由程序员代为处理。那程序员还有做自己工作的时间吗？等到有时间做自己工作时候已经是傍晚以后了。“啊，今天又要乘末班车回家了！”相信经历过这样的恶性循环的人不在少数。笔者就能拍着胸脯说“我经历过很多次！”，因此希望能有所改变。

用数据库来共享信息

即使是少数人参与的开发环境，为了顺利地推进开发也需要制定一些规则。笔者的公司就有这样的规则：

“无论什么样的情况，都要先把数据保存到数据库（以后简称DB）中。”

以下简称为“DB保存规则”（图2）。

图像和影像数据并不保存在DB中，而是通过Subversion来管理。但是其文件名等信息会保存在DB中进行管理。并且如果将数据的状态即未着手、制作中、完成等几种状态一起保存的话，非数据制作人也能了解当前的状况。

▼ 图2 DB保存规则

DB 保存规则
将所有能保存的信息都保存在 DB 中

1. 演出内容
2. 影像名
3. 台词
4. 声音数据
5. 各类消息
6. 数值数据
7. 影像数据的播放形式
8. 影像、图片、声音的文件名以及标示名
9. 各种数据的制作状态
X. 其他任意想到的内容

也可以使用Excel来确认和保存数据。在开发通过使用普及程度较高的Excel，使不太使用电脑的人员也能较好地理解。在工作表中准备好提交数据用的“发送”按键，并注明“数据编辑完成后按此键提交”，这比起长篇累牍地说明操作的细节，不是更好吗？

具体来说，Excel通过ODBC[①]和DB相连接。虽然如此，为了避免修改Excel时，导致DB中的数据被频繁地修改，一般会另行制作脚本来提交数据。

通过与DB间的这种便利数据交互，大家都可以通过Excel来确认数据的状态了。策划和管理者想知道某段影像是否制作完成，通过Excel就能方便地确认。实际工作中可能还是要直接询问具体的负责人，但是在询问之前预先查看一下DB中收录的数据的话，就能对进度有大致了解。通过确认数据还能得知设计师是否忘记提交数据。通过多人次的确认来确保没有数据遗漏，以此改善项目的品质，提高开发效率。

关于开篇提到的制作测试数据，因为所需的数据已经整理并存储在DB中，所以一旦需要的时候，就可以非常方便地使用了。例如，将播放影像时需要的视频和图片列表嵌入到程序中，只需要用SQL语句从DB中筛选出数据，转换成代码文件的形式并保存就行了。并且通过制作脚本，数据更新后只要运行脚本就能同步更新代码文件。更进一步，如果每次数据更新时，可以省略手动去运行脚本的话，那效率不就更高了？

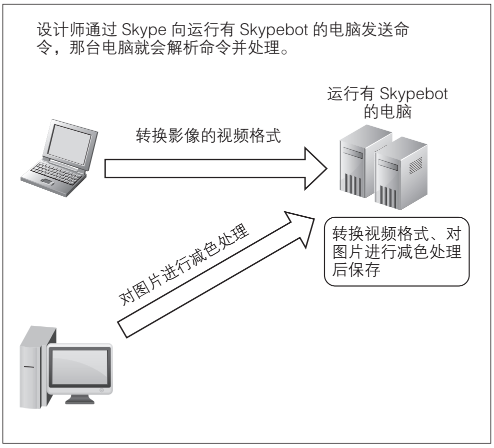
使用Skypebot自动处理

除了DB保存规则以外，公司内部一直使用一款通过接受Skype命令进行作业的优秀工具——Skypebot。什么是Skypebot？ Skypebot常驻于IRC的特定频道中，通过分析接收到的文字信号进行自动作业，并且是由Skype驱动的，所以命名为Skypebot。使用Skype聊天工具向运行有Skypebot的电脑发送文本指令，就能驱动电脑作业（图3）。

Skypebot如图4所示进行处理工作。

虽然Skypebot使用方便，但还是多少有一

▼ 图3 Skypebot的使用方法



① Open Database Connectivity，开放数据库连接。

些问题。要向Skypebot的处理中增加新功能时，就需要改动bot端调用的程序或脚本。在既定处理流程中插入额外处理的话，会造成大规模修改。因为是日常使用的工具，一旦修改可能造成很多奇怪的问题，还是比较劳心费神的。

Jenkins导入研究

Jenkins是持续集成工具，可能你还是无法理解，再简单点说就是能够做到“自动化测试，并统计结果”，或者“根据需要部署编译结果文件”，再或者“自动制作数据”。笔者想做的是尝试用Jenkins来替换Skypebot。

公司内大家对Jenkins都有所耳闻，但是在已经有Skypebot的情况下，导入Jenkins的目的是什么，有什么作用？导入Jenkins的费用以及其对应的效果，在现在的情况下难以估计。Skypebot现在也工作得很好，节省了大家很多的工作量。程序员不必每次都进行说明，只要记住“用Skype发送命令”这一步骤就可以了，即员工的培训成本很低，这也是Skypebot的魅力之一。

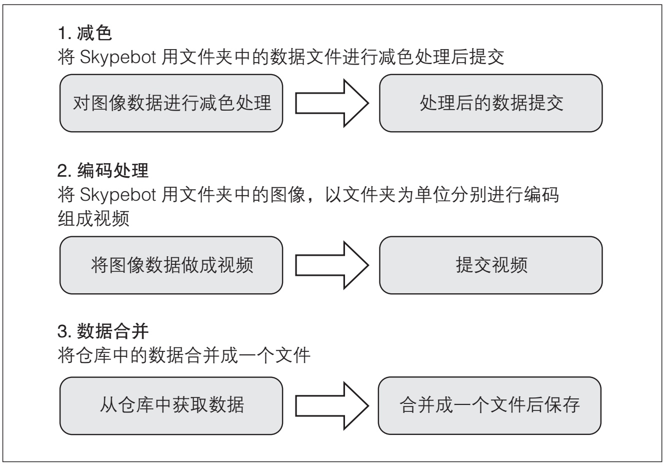
即便如此，笔者还是决定替换成Jenkins，除了之前提到的Skypebot存在的问题之外，还因为笔者以前的公司曾利用数台Jenkins，从服务器和同一楼层内数十台电脑，搭建分布式编译环境，通过实际大规模运用和管理Jenkins环境，笔者学到并理解了很多Jenkins的用法。虽然现在开发的规模变小了，但是笔者所做的和想做的事情并没有改变。另外从公司角度来说，导入Jenkins是在现有基础上，进一步探索提高开发效率的大好机会。

笔者对于现有业务如何使用Jenkins考虑了很久，决定先从Jenkins的导入工作入手，导入之后再研究如何将实际的开发工作和Jenkins联系起来。终端用户保持原有的操作方法，并从中找出能够改进之处，在开发人员的帮助下，通过实际的开发作业将Jenkins的使用方式具体化。

一旦导入Jenkins，普通的办法可能行不通

如何用Jenkins替换Skypebot？首先运行“将大量影像数据合并成一个文件”的工具，新建一个名为DATA_Compile的Jenkins的

▼ 图4　Skypebot的处理内容

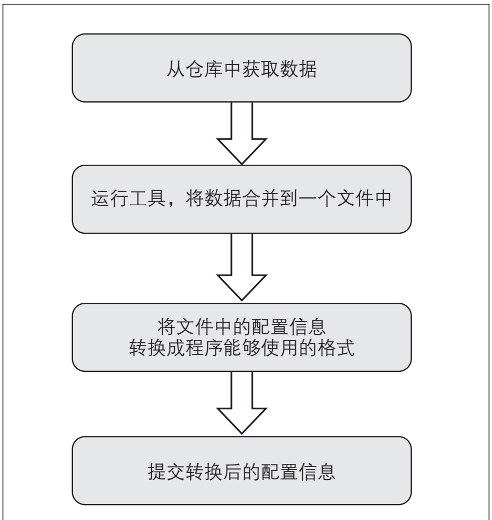


JOB，DATA_Compile将工具做成的数据文件作为JOB的输出并保存，并且将合并数据过程中由工具生成的配置文件转换成其他格式后提交到Subversion的仓库中（图5）。

总的来说，DATA_Compile JOB的制作相当麻烦。主要是Windows环境的硬件提供商没有提供命令行版本的工具。可能是因为GUI的工具使用起来比较方便，命令行工具非常少见，因此只能让Jenkins调用GUI的工具。经过了多次尝试、失败之后，总算搭建起了能够运行的环境。笔者会将在下回连载中，介绍这些问题是如何解决的。

刚解决一个问题，其他问题又接二连三地出现。配置文件格式转换完成后就是提交文件了，这一步也相当麻烦。在原始文件基础上进行更新还算简单，之后只要提交就能将更新后的文件保存到仓库中。但是如果更新的时间不凑巧就容易产生提交冲突，这时候就只能手动作业了：将手头的文件和仓库中的文件对比合并后再提交。这是多人共同开发环境下，时不时会发生的情况，虽然维护起来比较麻烦，但也没有办法。

▼图5 转移至Jenkins的DATA_Compile工作流程



你可能会问新建的文件要怎么处理。的确，新建文件的时候，必须先添加到仓库后再提交。提交的时候自动添加文件不是更方便吗？这样就有可能将不需要的文件错误地保存到仓库中。解决方法是通过制作脚本，提交前检查仓库中是否存在该文件，没有的话添加后再提交。提交的问题总算解决了，其他细小的问题又相继出现。现在使用的环境就是在这样逐渐解决问题的过程中搭建起来的。

环境构建好之后，使用初期可能会有各种明显的BUG。这是无法避免的，只有通过不断及时地修正使环境趋于稳定。运行几遍JOB，简单的BUG很快就能修正。所以要在周围同事的帮助下反复进行修正和测试。这样Jenkins的环境差不多就搭建完了，项目也随之渐入佳境。这时再出现问题的话可以通过添加处理脚本、拆分JOB等方式迅速处理。

JOB相关

在介绍使用Jenkins的事例之前，预先说明一下基本的JOB相关的操作。做好的JOB添加到Jenkins后如图6所示，在Jenkins的主画面（Dashboard）中以列表的形式排列。JOB名左侧的“S”列，用图标的形式表示着最近一次运行的结果：运行失败的话就显示为红色的圆点。圆点右侧的“W”列用气象图标的形式表示了最近一段时间JOB的状态，多次运行失败后就显示要变天的气象图标了。

单个JOB或者是多个JOB串联都能够正常运行。你可能会考虑用1个JOB来完成所有的

▼ 图6　主画面上显示的JOB列表



处理，但笔者推荐用多个JOB串联，每个JOB实现一个功能。因为用单个JOB来执行所有处理，如果中途处理失败的话，那么从开始到失败时间点的时间就都浪费了。具体来说，例如整体的处理流程是编译程序→整理数据→将程序和数据合并并压缩成zip文件→将zip文件上传至ftp，如果在最后一环上传至ftp时，由于ftp服务器问题导致上传失败，事实上只需要重新上传即可，但若是单个JOB的话只能从头开始重新执行。只需要修正脚本就可以了吗?如果忘记将失败的状态还原，可能会出现旧文件堆积或将旧文件上传至服务器等各类问题。另外根据切分方式的不同，整体实现的难易程度也会随之变化。

在导入Jenkins之前，把指令发送给Skypebot后，需要根据Skype的输出结果进行下一步处理。使用Jenkins后，串联的JOB会依次执行，所以就不必留意中间状态和结果。中途出现错误的话确认问题点，重新执行失败的JOB即可。不必一直盯着Skype看，这就大大减少了精神上的负担。

Jenkins不仅可以根据上一个JOB的运行结果启动不同的JOB进行后续处理，而且还可以设定在处理失败时发送通知邮件，使用免费的插件来解析log内容并进行相应的处理，分析数据自动绘制图表等。这部分的内容将在以后的连载中给大家介绍。

本回总结

以上大致介绍了作为本期连载涉及的开发环境以及Jenkins的相关用法。在JOB的制作和关联方面，日常的工作中，还需要根据实际情况进行修改。如何细分处理，如何将JOB串联运行，这些都会影响到处理的速度。要彻底搞清楚这些，需要一定的努力。

笔者还要啰嗦一遍，如果在处理的一开始就出现错误，那么多个JOB和单个JOB可能没有太大的区别。实际上所有处理放在单个JOB中，这样等待的时间将会很长。一旦JOB失败造成时间浪费，会产生很强的挫败感。因此在

项目的前半程，应该趁开发之外还有剩余时间，多设计一些实验用JOB，并进行反复试验。怎么做是可行的，怎么做是不可行的，只有在反复的尝试→失败→尝试的过程中去摸索和积累。因为无法预料什么时候会出现问题，所以要做好应对困难的思想准备。

还有，笔者所在的娱乐系统开发环境中，大家还不习惯使用DB。不习惯的原因多半是“太难了”、“跟我没有关系”等。但是数据的统一化管理能够大大缩短工作时间、减少工作量。

像Jenkins这样自动化的工具让人觉得很不可思议，所以经常听到这样的质疑：“这样做出来的东西可信吗？”但是对于程序员来说，比起单调的数据制作，让机器来制作数据，以此确保自己有更多的时间来编写程序不是更好吗？综合考虑程序员自身的工作量和时间，使用Jenkins切实能够为我们赢得更多时间来从事别的工作。

即便是使用DB统一管理数据、使用Jenkins实现自动化，在项目的最后阶段还是难免需要加班，但是没有出现被迫周末来加班的情况。即使有一些额外想做的事情，也不需要周末加班。托Jenkins的福，大家时间上宽裕了很多。

本期和大家探讨了将Jenkins引入笔者的开发环境的有关话题。从下期开始，我将介绍搭建环境中使用到的工具和脚本等。

安全实践鬼手诀

密码的重新思考



铃木弘信
suzuki.hironobu@gmail.com　译/芳龙

围棋的初学者都是先学习基本的定式（固定的下法），然后根据不同的局势情况灵活地应用这些定式，这样棋艺就会越来越高。而且，这些定式并不是一成不变的。

安全性也是一样的。学习基础知识当然很重要，根据当前的技术和发展趋势重新审视这些知识也很重要。本连载的第一期，我们先来谈一谈“密码”。

密码究竟是什么呢？

严格来说，应该这样问：“用密码的认证方法是指什么？”密码存在这样一种机制：先共享信息，然后通过确认是否持有这些信息来验证对方身份的正当性。其中那些“共享的信息”就是密码。

例如，《忠臣藏》里赤穗浪人杀入吉良家时，就是用“山”和“川”作为共享信息来进行确认的。这也可以称为一种密码认证。此时，“山”和“川”就是密码。

但是，这并不是本人认证，而只是确认是否是共享信息的持有者（这里用的是“者”这个词，但“持有者”也可能不是人类）。有时胡乱尝试一个密码，碰巧撞对了，这虽说是偶然的，但也能被认为是“共享信息”的持有者。

密码：实现通行的“码”

分析计算机的密码实现方法，从表面上看大概是这样的：

先让用户输入8个左右的字符，然后和事先保存在计算机内部的信息进行对比验证，如果相同的话，就可以共享隐匿的信息。

密码中字符串的长度，以及字符的可用类型，都是由系统决定的。这方面没有明确的标准，一般由具体的实现而定。比如登录UNIX时使用的密码，可以是大小写英文字母，0～9的数字，#、%等几种特殊字符。但是，网站的登录密码大多只使用大小写英文字母和0～9的数字。以前也有不区分英文字母大小写的电脑密码。这样看来，即使密码可用字符长度只为1，其实现方法也依赖于系统[①]。

密码：希望通过验证的“码”

以下2个密码经常被作为密码安全的反例。

① Hironobu1963

② 12345678

①“安全性策略”说起来好听，但目前很多安全系统都是马马虎虎，应付了事。

①是名字和出生年份的组合。这样的密码本人是不会忘记了，但别人也很容易推测出来。

②不管是谁都会觉得有些过分简单了。但现实还真有人就用这样的密码，你可别笑。2012年7月，CNET上登了一则有趣的报道：Yahoo hack reveals most-used passwords[②]。这则报道公布了2012年从美国雅虎泄漏出来的45万个密码的分析结果。表1列出了笔者比较感兴趣的一些密码。

表1　美国雅虎45万个密码的分析结果（部分）

发现的密码数	此类密码
2295	12345, 123456, 1234567...
160	111111，0000000，777777...
780	password
233	password1，password2
106	batman，superman...
27	ncc1701，ncc1701a...

有2295个密码是像12345...这样从1开始顺序数列的数字串，有160个密码是像111111这样相同数字重叠的数字串，有780个密码是password这个单词，有233个密码只是在password后面加了个数字，有106个密码是像batman、superman这样谁都能想到的名字。不仅是笔者，各位读者也会觉得这样的密码简直太简单了，但是这就是现实中屡屡被使用的密码。仅仅这些就有3000多个账号可能会被恶意黑客不正当使用。

另外27个像ncc1701、ncc1701a这样的字符串是什么呢？笔者看到这些密码的时候，才恍然大悟。实际上，这是《星际迷航》中U.S.S.联邦星舰“企业号”的登录号。这些人想使用这些密码的情怀是可以理解的，但是，如果笔者都能一看就知道的话，必然已经在维基百科上存在具体记录了。把维基百科的文本存档下载下来，抽出所有的字符串模式，进行排序，然后把重复的字符串去掉，这些密码肯定会在剩下的结果中。虽然维基百科上有大量的数据，但是把这些数据抽取出来，现在也不是很难的事情。用笔记本电脑就可以做基于维基百科的密码类推字典。

PAM和libpam-cracklib

GUN/Linux的密码认证现在采用PAM（Pluggable Authentication Modules）机制。这比古典的UNIX有更高级的密码认证框架。

PAM正如其名，以插件的形式存在，在必要的时候可以对密码检查进行设置。

例如Debian GNU/Linux 6.0.7，没有装libpam-cracklib插件时，可以容许6个字符的密码。装了libpam-cracklib插件后，默认设置就是8个字符。在有些系统下多于8个字符的部分会被忽略（笔者还没遇到这样的情况，但是这种情况是有可能存在的）。如果使用libpam-cracklib，密码的最小长度可以达到12个字符。libpam-cracklib的安装方法如下所示：

```
# sudo apt-get install libpam-cracklib
```

安装完成之后，在/etc/pam.d/common-password加上下面这行，密码的默认设置就变成8个字符了。

```
password requisite  pam_cracklib.so ↵
retry=3 minlen=8 difok=3
```

② http://news.cnet.com/8301-33692_3-57471417-305/yahoo-hack-reveals-most-used-passwords/

没有发挥应有作用的密码

上述报道并不是说泄露的密码数据都已经被破解了，只是列出一些通过穷举法可以找到的密码。那么到底有多少密码可以被破解呢?

报告指出，泄露的45万个密码中的137 559个已经被破解，约占全体的30%。而且，其中106 873个是和Gmail、Hotmail通用的密码。

就在笔者撰写本文时，传出了迪诺斯（Dinos）公司遭到111万次非法访问，1.5万次非法登录的新闻③。这次针对迪诺斯公司的大量非法访问，和刚才说的密码泄露事件一样，都是利用其他网站上泄露的密码进行的尝试登录。

最近，很多人都用邮箱地址作为账户名，如果在不同的网站使用同一密码，一旦一个网站的密码被破解，入侵者顺藤摸瓜，就可以大摇大摆地进入其他网站的账户。

密码本来的目的是为了识别用户，但是，系统中大约3成的密码都没有起到这样的作用。而且，这些不负责任的密码所带来的影响还波及那些根本没有泄露信息的网站。

在迪诺斯事件中，1.35%这个百分比不可小觑。在迪诺斯公司并未有任何安全性泄露的情况下，可以被非法使用的账户居然多达1.5万个！这么多数据如果被恶意使用，其后果不堪设想。

可能有人觉得只要设置密码就可以高枕无忧了。但是看看这些触目惊心的数字，可见迄今为止的密码认证并没有达到预期的效果。

③ http://itpro.nikkeibp.co.jp/article/NEWS/20130510/475982/

系统中适当的密码管理是指什么

我们重新来整理一下密码的管理方法以及非法破解密码的方法，便于理解和把握整个论题。

但是，像之前提到的123456、password这样的密码，或者账户名是admin、密码也是admin的情况，只要试几次就可以破解，根本就不需要我们接下来要介绍的密码破解手段。使用这样的密码只能说是太愚蠢了。不过，这也不能全怪用户，笔者认为系统能容忍这样的密码本身就是一个欠缺。

现在开始进入主题。我们讨论的前提是：对于有一定复杂度的密码，密码管理文件从网站泄露，并使用专用计算机破解这些密码。关于入侵网站的方法以及数据文件的泄露等将在以后的章节进行讨论。

先说明以下3种密码管理模式，然后介绍一下都是采用怎样的方法进行密码破解的。

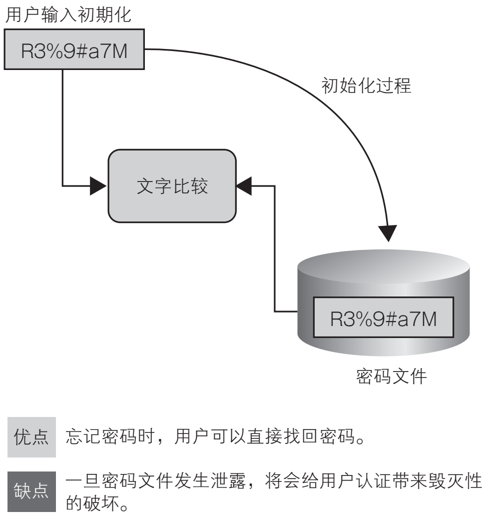
没有保护措施的密码管理

很多人都认为，所谓的密码认证就是把用户输入的字符串和已经保存好的密码进行比较（图1）。

这种方式很容易实现，但是，一旦密码文件泄露，就关乎服务能否存续，整个系统的用户认证也会崩溃。这是因为密码不用经过任何加工就可以直接拿来使用。

也有人认为控制好密码文件的访问就可以确保安全性。然而，只有这种保护的话，就没有考虑到信息被泄露的可能性。系统仍处于脆弱的状态中，一个错误就可能导致整个系统瘫痪。

◆ 图1　保存的密码没有受到保护



这样的系统存在很大的问题，即使我们不强调很多读者也能理解。我们可能乐观地认为这样的系统不是很多，但是，用户忘记密码时，可以亲切地告知原密码的系统，在我们周围并不少见。

引入单向散列函数

单向散列函数是指具有以下特性的函数：对于值x，使用函数H算出值H(x)，从H(x)很难推算出x。单向散列函数也不是只有MD5、SHA-1、SHA256这些。使用加密函数产生的消息认证码，也可以作为单向散列函数使用，比如古典UNIX的密码，就是使用DES加密函数的DES-CBC-MAC消息认证法生成。

这里经常使用“对密码加密”的表达方法，但是这种说法容易让人觉得可以加密，是不是也可以解密呀。事实上，这种算法是不可逆的，所以才叫单向散列函数。

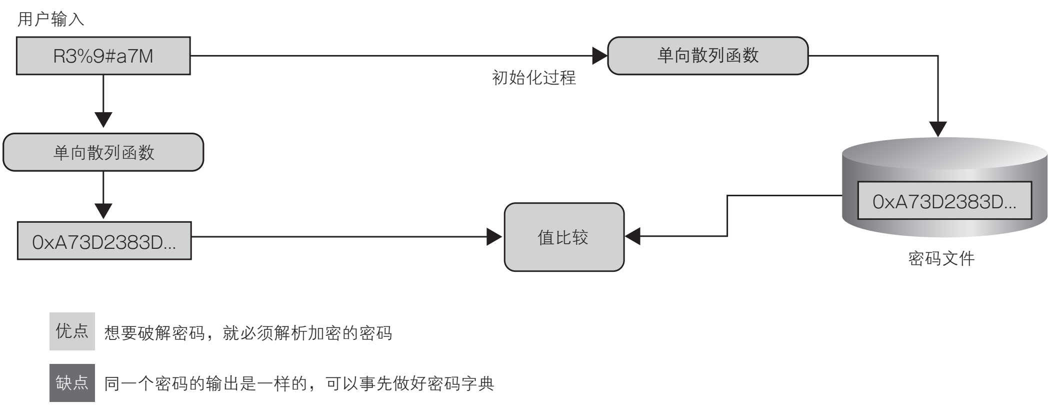
先输入密码，然后使用单向散列函数算出其哈希值，再与事先保存好的单向散列函数算出的哈希值进行比较（图2）。比如Windows XP等使用的LM哈希（LAN Manager hash）就是这种方式。

与图1中直接保存不经过任何加工的字符串相比，已经有很大进步了。但它还是有缺点：相同的输入会得到相同的输出。所以，如果使用的是在字典中可以查到的密码，那么事先处理一下，就可以做成哈希值的字典。也就是说，这样的密码系统对字典攻击的抵御是很脆弱的（请参照：密码强度）。

加盐（Salt）的密码管理

只有单向散列函数的话，可以使用攻击字

◆ 图2　引入单向散列函数



典破解。解决这个问题的办法就是加盐：为每个用户加一个不同的随机值，然后再使用哈希函数（图3）。

盐即使不隐藏起来也没关系。盐越长，用于破解的攻击字典就越庞大。盐虽然是随机数，但也不需要太大，只是保证当前几年的安全性的话，有32比特（4字节）就足够了。实际上70年代使用的是12比特左右。考虑到将来的话，有80比特（10字节）就可以说相当安全了[4]。

UNIX的密码处理，是要多次使用单向散列函数进行计算的一种比较费时的方法。但这种方法对于同时要应对大量用户的Web网站而言，认证服务器可能就会超负荷。所以是否要采用这种方法，还是需要慎重考虑的。增加密码复杂度，可以采用增加可用字符种类，或者增加字符数等重要方法。这才是普通的密码管理[5]。

即使是安全的密码

提到安全的密码时，通常总是推荐这样的密码：组合复杂的密码、尽量长的密码、频繁更换的密码。但是，笔者对此有一些不同的看法。

●组合复杂的密码

人们会加入自己的偏爱，不能确保安全的随机性。

●尽量长的密码

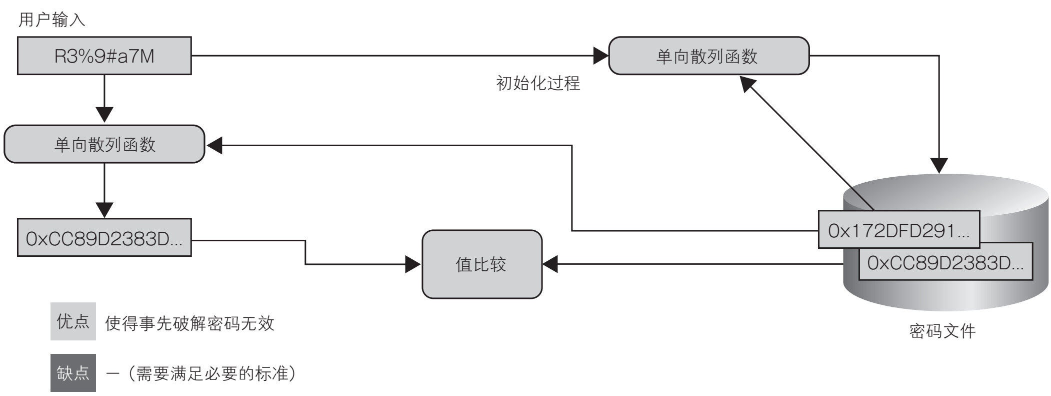
高估了人类的记忆力。结果还是倾向于选择比较容易记忆的密码。

●频繁更换的密码

上述两种情况的重复。结果就很可能出现容易记忆的密码，或者是同一个密码重复使用的情况。

有的网站会有这样的情况：比如将“I love you”按一定的规律换成“1 l0v3 y0u”，这种不在英文字典上出现的字符串就可以避免字典攻击；或者是选择自己喜欢的句子的各个单词的头文字，比如“I want to eat cake”就成了“iwtec”。但这种杂乱程度对电脑来说根本不算什么。

◆ 图3　加盐



④ http://news.cnet.com/8301-33692_3-57471417-305/yahoo-hack-reveals-most-used-passwords/

⑤ 这里所说的“普通”不是指单纯在统计上种类增加，而是指与“理想接近”的普通。（参考：Dream Fighter by Y.Nakata）

对于从互联网可以自由访问的网站服务而言，如果只靠密码来防御入侵的话，所有的密码只能是用机器生成的足够长的随机字符串才行。当然，数十位甚至上百位的随机字符串，人们是记不住的。所以就要使用密码管理工具。与使用计算机可以轻易破解的密码相比，把足够长的随机字符串写到纸上，好好保管那张纸，这样的笨方法反而更安全。

或者，干脆彻底放弃，想一个自己下次登录也记不起来的、别人就更记不住的超复杂的密码。等到自己使用的时候就重新设置密码。

不这样走极端的话，本来赋予密码的用户认证功能就实现不了。

◉密码强度

使用不在字典上、完全是随机字符串的密码，到底可以安全到什么程度呢？前提条件是使用英文字母（区分大小写）、数字（0–9A–Za–z）的62个字符。考虑Web服务标准的密码模式，8个字符时，约有218兆3401亿个；10个字符时，约有84京个。

假设使用的单向散列函数是SHA-256，可供使用的计算机有一台，其配置是在秋叶原可以搞定的水平，花多少钱暂且不管，插4条经常玩大型游戏的人才用的NVIDIAGeForce GTX-690，装4核以上的CPU。

SHA-256在1秒内大约计算22亿次。顺便提一下，SHA-1的话大约是44亿次，MD5的话大约是140亿次。不过说到底这只是笔者的大概估算[①]。

因为是求平均值，我们只计算8个字符组合218兆个的一半109兆，大约是49 622秒，也就是大约有14个小时的话就可以破解。10个字符的话，大约要6年多的时间。

在没有使用加盐的单向散列函数的情况下，可以使用逆推字典。8个字符的情况，产生逆推字典的计算时间如上所示。但是要保存这些数据，这次假定的条件是实现不了的。别的什么都不管，只做字典的话，因为要保存大量的密码和哈希值的对应关系列表，大约估算一下也需要10 PB（petabyte）的容量。这样大的容量可以和数据中心使用的最大级别的外部存储装置相匹敌了。

虽说现在硬盘容量又大价格又便宜，那它的容量也不过也就是4 TB左右。当然以后还会发展，但是要想在秋叶原就买到PB级别的存储设置，还得是很久以后的事情。

但是，如果使用加密学方法[②]，并不需要保存所有的对应关系，只需做一个辅助查找的表。虽然数据交换会花一点时间，但需要存储的数据量会发生戏剧性地减少。而且，实现方法已经公开[③]了。使用这些代码，结合实际系统的情况使用起来并不是一件很难的事情。

这里顺便提一下，用于Windows XP和Vista的字典表已经做好了。[④]对Vista而言，8个字符的随机英文数字（区分大小写）密码，表的数据量大约是134.6 GB（检索成功率99%）。这已经是笔记本电脑可以承受的容量了。也就是说笔记本电脑也可以在几秒或几十秒的检索时间范围内，以99%的准确率来破解密码了。

① 这是根据Ivan Golubev' s Password Recovery Suite推导出来的。http://www.golubev.com/igprs/
② Making a Faster Cryptanalytic Time-Memory Trade-Off，http://lasecwww.epfl.ch/~oechslin/publications/crypto03.pdf
③ OPHCRACK网站 http://lasecwww.epfl.ch/~oechslin/projects/ophcrack/
④ http://ophcrack.sourceforge.net/tables.php

总结

现实中，不得不说很多地方的密码管理和运用方法都已经Out了。然而，换个角度看，目前密码还处于全盛期。

笔者认为应该积极采用公用密匙方式的用户认证方法。这种技术对UNIX用户来说并不陌生，因为有SSH公用密匙认证等。但就现状来说，一般情况下还是很难普及。

尽管现状如此，一些高科技企业的认证方式还是正在慢慢地发生着变化。比如Google账户已经有双重认证了。笔者就在用，并没有想象中的那么复杂。笔者认为，像银行交易等涉及金额可能很大的场合，使用一次性口令的安全令牌也是很合理的方法。

现在可以说正处于从古典密码认证到新一代认证的青黄不接之际。在很长一段时间内，还会继续使用现在的密码。

关于用户认证的讨论还远远不够，我们下回再分解，这次就到这里。



菜鸟编程入门

开发一个iPhone阅读类应用

第3回

准备APP开发需要的图片

GimmiQ（GimmiQ；Itanokumanbou&lio&reverse）
http://ninebonz.net/
http://www.studioloupe.com/
译/苏祎

本连载希望从未编程过的人也可以体验制作APP的乐趣。此次我们将介绍制作APP所需要的图片的制作方式。

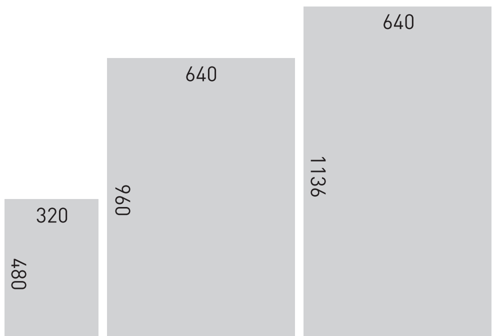
本次主题

制作iPhone APP需要各种准备，其中一个即为图片的准备。这与程序一样，是完成APP制作中非常重要的一个环节。本次，我们将介绍制作APP所必需的图片种类，并解说准备这些图片所有的步骤。

为何需要准备各种图片？

由于不同的机型要使用到的图片会有所不一，所以我们需要准备多种图片。现在，仅iPhone和iPod touch就有如下3种机型：（参见图1）

▼ 图1　3种尺寸的屏幕解析度



1. 从第一代iPhone到3GS为止的3.5英寸显示屏机型（分辨率为320×480像素）
2. iPhone4到4S为止的3.5英寸Retina显示屏机型（分辨率为前者2倍的640×960像素）
3. iPhone5开始的4英寸Retina显示屏机型（分辨率仅高度有改变，为640×1136像素）

iPhone5刚推出时，4英寸界面适用的图片是作为推荐的，但并不是必要条件。然而，今后非适配4英寸界面的APP将无法通过审核，所以这就变成了开发者必须对应的内容。在本连载中的第1回和第2回中，为了能简洁明了地解说APP的开发，只适配了3.5英寸的界面，从本文开始，我们将也应对4英寸的界面。所以至少需要准备3种机型的图片。

制作图片

如前文所述，APP开发需要通过各种样式来制作图片，但简单来说实际上只需要关注3.5英寸和4英寸两种尺寸就行了。旧机型的3.5英寸（低分辨率）所需要的图片，仅需要通过将高分辨率的图片按照原本的50%（一

半）缩小就可以了。这个只要通过Mac内置的APP“预览”（图2）就可以简单完成。

首先，将希望缩小的图片一起拖拽到预览中打开。这样在左侧就会纵向显示图片的缩略图（只有1张时不会出现缩略图）。选择其中一张缩略图，按住command+A键选择所有图片，从菜单栏的“工具”中选择“调整大小”（图3），在弹出的对话框中，在宽度和高度的右侧的下拉框中选择百分比，并将两个数字都从100调整为50（图4）。接下来点击“OK”，即可把所有图片缩小到一半的尺寸。

▼ 图2　预览的图标



▼ 图3　预览中显示图片



▼ 图4　预览中将尺寸变为50%



图片文件的格式

iPhone中可以读取各种格式的图片文件，但苹果公司最为推荐的是PNG格式的图片，所以请尽可能使用PNG。PNG也可以成为透明（半透明）的图片，所以在APP开发中比较易于使用。比如说，使用圆形按钮时，若是PNG的话，将余白部分保持为透明就可以简单地和背景融为一体。（图5）

图片的像素尺寸要使用偶数

在制作图片的时候，请注意保证像素数字为偶数。像素最少以1像素为单位，不存在半像素。如果在Retina用的图片尺寸中使用了奇数像素的话，因为不存在刚好一半的尺寸，所以在制作低分辨率机型适用的图片时，就会自

▼ 图5　透明图片的效果



动被变换成最近似的像素。这样的话，就有可能会造成设计上的误差，所以需要注意。

图片的种类

APP开发所必备的图片如下：

【图标图片（3种）图6】

- icon.ong（57×57像素）
- icon@2x.png（114×114像素）
- APP Store用的高分辨率图标（1024×1024像素）

【默认图片（至少一个）】

- Default-568h@2x.png（640×1136 像 素的图片）

【APP内要适用的其他图片（3种）】

- 3.5英寸屏幕，非Retina图片（320×480像素，或与480×320像素适配的）
- 3.5英寸屏幕，Retina图片（640×960像素，或与960×640像素适配的）
- 4英寸屏幕，Retina图片（640×1136像素，或与1136×640像素适配的）

▼图6　3种图标



接下来我们逐个详细说明。

@2x图片

最先需要说明的是，在文件名和扩展名之间有@2x的图片文件。这个是为了区分高分辨率“Retina屏幕用”所使用的画面。只要在文件名后加入@2x，程序就会自动将其识别为Retina用图片，并在使用高分辨率的机型时自动切换。所以在低分辨率和高分辨率的机型间的图片切换不需要制作特别的程序。

相反，3.5英寸和4英寸的图片切换只能通过程序进行。所以在图片命名上并没有特别的规则。在与Default图片一样的文件名后加入代表568磅的-568h或者是加入-4inch就更容易管理了。

像素和点的区别

此次用到很多数字，所以希望大家务必明确，像素（pixel）和点（point）的区别，避免混淆。程序在描述或移动文件时，不使用像素数而是通过点数来进行计算。这个点数在标准解析度（非Retina）画面中会按照1点=1像素（横竖1×1像素）来计算，所以点数和像素数是一种比例关系。但是在解析度为2倍的Retina屏幕下，这个点数的处理需要统一，所以在Retina显示屏机型中，横竖2像素会计算为1点。就是说，在Retina显示屏下，点是像素的一半。在新的4英寸显示屏中，纵向有1136像素，但换算成点的话，就是其一半，是568点。4英寸的图片文件名后面的“568h”就是指这个点数。

由于点数不是常用数据，你可能需要一点时间来习惯，但实际上这个并不复杂。在开始

编程后自然而然就会理解，所以现阶段不需要对此太过深入思考。

图标图片

我们一共需要准备3种图标图片，分别为两种在手机桌面排列的图标图片（低分辨率和高分辨率）以及App Store所需要的高画质图片。

App Store用的高分辨率图标（1024×1024像素）由于不在APP中使用，所以不需要放入到APP中。但是APP提交审核时需要上传，所以一并制作完会好一些。最简便的方法是，先制作1024×1024像素尺寸的图标，将其缩小成114×114像素和57×57像素两种图标。但是由于缩小率很高，所以缩小后的效果可能和预期不一样。这时就需要根据各个尺寸进行微调，或是从头开始制做。特别是57×57像素分辨率很低，所以很精细的图案不易表现出来。

另一点需要注意的是，图标图片是不可以使用半透明背景的。所以图标请按照正方形来制作，四角的圆弧会在陈列到手机桌面时自动被处理。（图7）

启动图片

启动图片是从点击APP图标瞬间，衔接至APP启动（完成读取）的短时间的图片。这也被称为启动画面。介于最近iPhone的性能来说，除非是启动时间非常长的APP，一般来说不会等待很久，所以直接使用一张黑色的图片也是可以的。但是，比如说要展示品牌的logo，或是显示为“启动中”等内容，也可以对启动画面做一些优化加工。

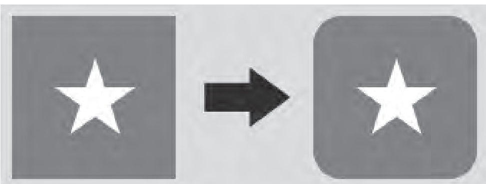
有一个技巧，就是将启动后最先出现的画面截图，并设置为启动画面，这样就会让人感觉到该应用启动很快。但对此做法，大家的意见不一。也有一部分用户反馈APP明明已经启动了，但触摸后没有任何反应，感觉到体验不佳。这种情况下，若将半透明的黑色图片（50%～70%左右）加在启动画面之上，让用户感觉到还没有完全启动，应该就可以避免这样的混淆了（图8）。

APP内使用的其他图片

APP内使用的图片，是指按钮图片、背景图片，若是写真集APP的话还包括照片的图片等。按钮图片等内容可能只需要准备低分辨率和高分辨率两个版本，但布满整个画面的背景图片，以及需要全屏展示的照片及图案等，就需要准备3.5英寸的低分辨率和高分辨率图片，以及4英寸的高分辨率图片共3种。

阅读到这里，你可能会觉得需要准备很多

▼ 图7　制作成正方形的话会自动将边角变成圆弧



▼ 图8　默认界面



图片，但其中大约一半图片都是可以通过压缩高分辨率原图得到，所以工作量并不是非常大。只要掌握好需要制作哪些图片，就可以有效顺利地完成图片制作。

我们以第2回连载中制作的写真集APP为例，具体地思考一下。需要制作的图片如下：

（a）在手机桌面会展示的图标两种

（b）启动图片

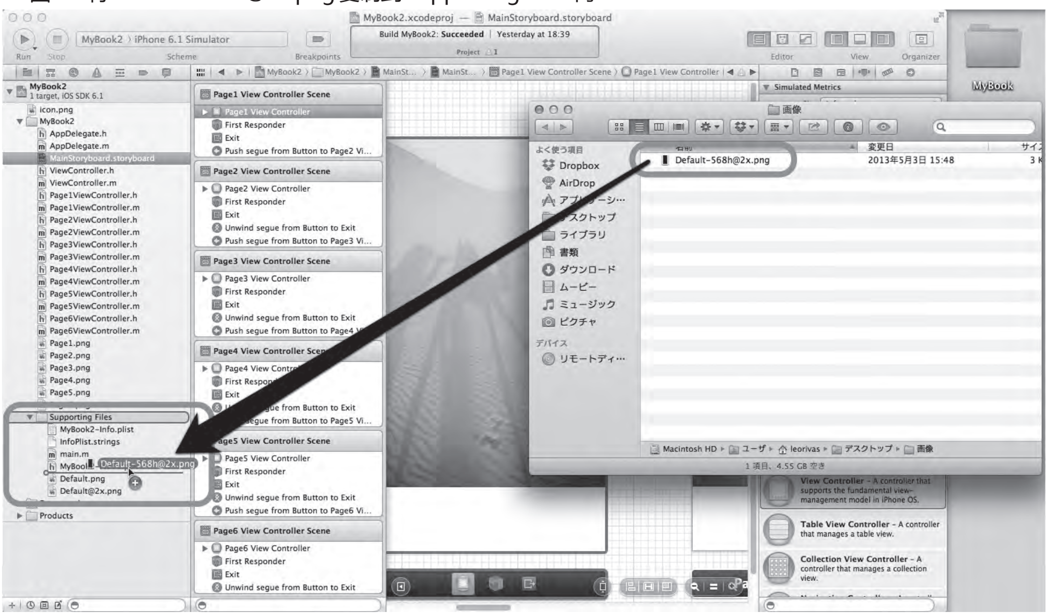
（c）各页面的照片图片各3种

（a）的图标现在还不需要向App Store提交，所以只要准备114×114像素和57×57像素两种就可以了。但如之前所述，若采用从大图片缩小制作小图片的方法，也可以从需要向App Store提交的图标开始制作。在本文的举例中，我们将两个图标文件命名为icon.png，icon@2xl.png。

（b）默认图片是在新项目启动时就应自动包含的文件（Default.png，Default@2x.png，Default-568h@2x.png）。但是在本连载中，第1回的第9步中特地删除了Default-568h@2x.png造成了不适配4英寸，所以仅需要重新添加这一张图片（今后制作新的APP时，已经存在的自动生成文件，不需要另外添加）。只要图片尺寸是640×1136像素，即使是全黑的一张图片也是可以的。

（c）中各页面照片的图片（Page1.png~Page6.png，各位制作的写真集APP里面所需要的图片张数应该会不同，所以此处请根据具体情况调整）之前都是适配3.5英寸屏幕，而非Retina屏幕，所以分别需要制作“4英寸屏幕的Retina图片”和“3.5英寸屏幕的Retina图片”。并将前者命名为PageX-568h@2x.png（X代表页面编号），将后者名命名为PageX@2x.png（X代表页面编号）。这样的话，要添加的图片准备

▼ 图9　将Default-568h@2x.png复制到Supporting Files内



完成，其文件名为Page1-568@2x.png~Page6-568@2x.png与Page1@2x.png~Page6@2x.png。

使APP适配4英寸屏幕的方法

在所有的图片准备完成后，就终于要用Xcode来进行4英寸的APP适配了。从这里开始，将以对第2回制作的APP的处理为例，来进行说明，所以请使用Xcode打开已制作完成的APP项目（在本文中的举例中命名为MyBook2.xcodeproj）。

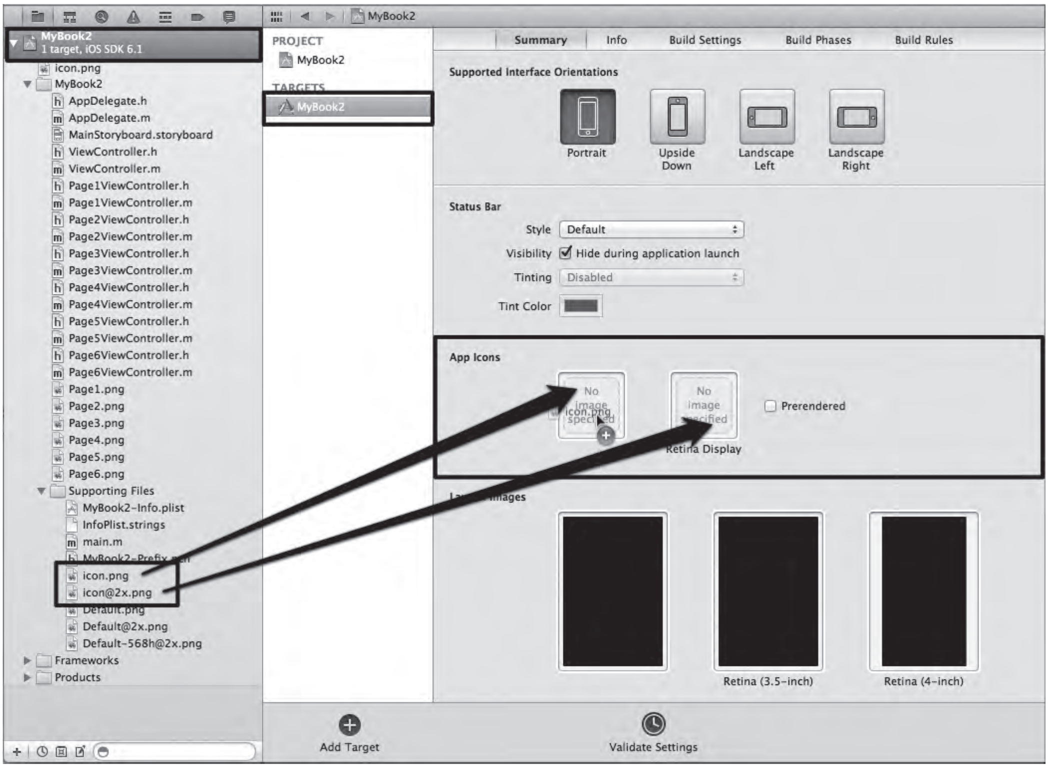
首先，请将PageX-568h@2x.png拖拽到Xcode的“Supporting Files”中（图9）。同理，将两个图标文件也加入到“Supporting Files”中，而将需要添加的照片图片加入到与原来的照片同样的文件夹（例子中为“MyBook2”）。

为了完成图标的设定，点击左菜单栏的最上方的项目文件，选择“TARGETS”下面的项目文件名（例子中为MyBook2，见图10）。滑动鼠标找到“App Icons”的区域，将刚才添加到“Supporting Files”的图标文件按照图11所示逐个拖拽到四方形的盒子中。左边是低分辨率的图标，右边是Retina显示屏的图标。这样就在手机桌面可以作为图标展示了。默认设置中，图标会被自动加上光泽感，但若不想要此效果，请将“Prerendered”勾选。

接下来，只要添加上Default-568h@2x.png就算是适配了4英寸的屏幕了，但里面的图片若没有调整成4英寸的屏幕的话，就仍然会有余白，无法算作是真正完成了适配。

在没有放入Default-568h@2x.png的状态

▼ 图10　将图标图片放到相应位置



下，（未适配4英寸）启动APP的话，图片会上下居中，并在屏幕上下部分别会出现黑色的条状（图11的左）。但若放入了Default-568h@2x.png后（适配了4英寸），通过模拟器确认的话，图片会顶端对齐，下方会出现较大的白色余白（图11的右）。虽然看上去不是很美观，但理论上这样算是已完成了4英寸的适配。

这之后，关于3.5英寸和4英寸的画面切换，处理的方法会不一样。最简单的、不需要写任何代码的方式就是在底部余白部分（88点）中加入一些设计，只有在4英寸的屏幕时会看到，3.5英寸的时候就会隐藏（图12）。

但是若希望在3.5英寸的时候和4英寸的时候分别显示不同的图片的话，就需要通过编程来处理了。

总　结

以上即为iPhone APP开发所需要的图片准备。在下一期的连载中会解说通过实际编程来判别不同机型，以实现图片在3.5英寸屏幕4英寸屏幕间的切换。

▼图11　4英寸默认图片



▼ 图12　对余白进行设计



如何构建超级系统管理程序 Hypervisor

正确理解虚拟化技术

第10回 用Intel VT-x构建超级系统管理程序之六 用户空间（user land）的I/O模拟

文/浅田拓也（ASADA Takuya）Twitter @syuu1228　译/杨驹武

引子

上期我们讲解了从VMX non root mode向vmm.ko发出VMExit时的处理。本期我们来讲一讲接收到由I/O命令引起的VMExit后，在用户区进行的模拟处理。

关于源程序

本连载将对FreeBSD-CURRENT下BHyVe的源程序进行讲解。该源程序可以从FreeBSD的Subversion文档库里取得。修订版本号为r245673。

在PC里安装Subversion后，用下面的命令取得源程序。

```
svn co -r245673 svn://svn.freebsd.org/
base/head src
```

/usr/sbin/bhyve实现的虚拟CPU执行处理概要

/usr/sbin/bhyve生成和虚拟CPU个数相同的进程，每个进程分别向/dev/vmm/${name}发出VM_RUN ioctl（图1）。vmm.ko接到ioctl后把CPU切换成VMX non root mode，执行客户机操作系统（VMEntry）。

VMX non root mode状态下，如果发生了需要超级系统管理程序介入的事件，控制会返回vmm.ko，引发VMExit事件。

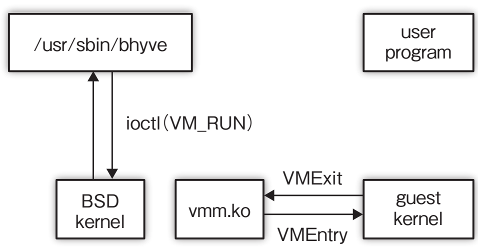
如果事件需要由/usr/sbin/bhyve处理，会返回ioctl，控制转移到/usr/sbin/bhyve。/usr/sbin/bhyve参照事件的种类和寄存器的值进行设备模拟等处理。

这次，我们来看一下/usr/sbin/bhyve的设备模拟处理部分。

/usr/sbin/bhyve的I/O命令处理

承续上期解说，我们再来看一下由I/O命令引起VMExit的情形。VMExit的相关信息被写入VM_RUN ioctl的参数struct vm_rum

▼图1　由VM_RUN ioctl执行虚拟中央处理器的图示



的vm_exit成员（struct vm_exit）中，ioctl返回时复制到用户区。/usr/sbin/bhyve接收到以后，参照vmexit->exitcode来判断VMExit的起因，根据VMExit的起因，调用不同的处理。由I/O命令引起VMExit时的exitcode是VM_EXITCODE_INOUT。

在VM_EXITCODE_INOUT时，通过struct vm_exit把I/O命令模拟所需的信息[端口号，访问带宽，写入值（非读入时），I/O方向（in/out）]传递给vmm.ko。

/usr/sbin/bhyve把这个值传递给I/O模拟器，通过I/O端口号判断要访问的设备，调用该设备控制程序。

设备控制程序执行结束后，/usr/sbin/bhyve再次发出M_RUN ioctl，客户机再度开始运行。

我们按照上面所讲，详细看一下源程序。代码清单1、代码清单2、代码清单3和代码清单4是源程序。

vmmapi.c和bhyverun.c的解说

libvmmapi是把访问vmm.ko的ioctl · sysctl抽象化的程序库。/usr/sbin/bhyve · /usr/sbin/bhyvectl调用它们来访问vmm.ko（代码清单1）。

代码清单2的bhyerun.c是/usr/sbin/bhyve的核心程序。

▼ 代码清单1　lib/libvmmapi/vmmapi.c

```
……（省略）……
280 int
281 vm_run(struct vmctx *ctx, int vcpu, uint64_t rip, struct vm_exit *vmexit)
282  {
283    int error;
284    struct vm_run vmrun;
285
286    bzero(&vmrun, sizeof(vmrun));
287    vmrun.cpuid = vcpu;
288    vmrun.rip = rip;
289
290    error = ioctl(ctx->fd, VM_RUN, &vmrun);      ← ① 在上期连载的最后返回到用户区的ioctl又回到了这里
291    bcopy(&vmrun.vm_exit, vmexit, sizeof(struct vm_exit));      ← ② 把从vmm.ko传递过来的vmexit信息复制后传回调用源
292    return (error);
293 }
```

▼ 代码清单2　use.sbin/bhyve/bhyverun.c

```
……（省略）……
294 static int
295 vmexit_inout(struct vmctx *ctx, struct vm_exit *vme, int *pvcpu)
296 {
297   int error;
298   int bytes, port, in, out;
299   uint32_t eax;
300   int vcpu;
301
302   vcpu = *pvcpu;
303
304   port = vme->u.inout.port;      ← ⑥ 展开VMExit发生时vmm.ko取得的I/O命令模拟处理所需的信息[端口号、访问带宽、写入值（非读入时）、I/O方向（in/out）]
305   bytes = vme->u.inout.bytes;
306   eax = vme->u.inout.eax;
```

```
307   in = vme->u.inout.in;
308   out = !in;
309
……(省略)……
322   error = emulate_inout(ctx, vcpu, in, port, bytes, &eax, strictio);
323   if (error == 0 && in)
324     error = vm_set_register(ctx, vcpu, VM_REG_GUEST_RAX, eax);
325
326   if (error == 0)
327     return (VMEXIT_CONTINUE);
328   else {
329     fprintf(stderr, "Unhandled %s%c 0x%04x\n",
330       in ? "in" : "out",
331       bytes == 1 ? 'b' : (bytes == 2 ? 'w' : 'l'), port);
332     return (vmexit_catch_inout());
333   }
334 }
……(省略)……
508 static vmexit_handler_t handler[VM_EXITCODE_MAX] = {
509   [VM_EXITCODE_INOUT] = vmexit_inout,
510   [VM_EXITCODE_VMX] = vmexit_vmx,
511   [VM_EXITCODE_BOGUS] = vmexit_bogus,
512   [VM_EXITCODE_RDMSR] = vmexit_rdmsr,
513   [VM_EXITCODE_WRMSR] = vmexit_wrmsr,
514   [VM_EXITCODE_MTRAP] = vmexit_mtrap,
515   [VM_EXITCODE_PAGING] = vmexit_paging,
516   [VM_EXITCODE_SPINUP_AP] = vmexit_spinup_ap,
517 };
518
519 static void
520 vm_loop(struct vmctx *ctx, int vcpu, uint64_t rip)
521 {
……(省略)……
532   while (1) {
533     error = vm_run(ctx, vcpu, rip, &vmexit[vcpu]);
534     if (error != 0) {
535       /*
536        * It is possible that 'vmmctl' or some other process
537        * has transitioned the vcpu to CANNOT_RUN state right
538        * before we tried to transition it to RUNNING.
539        *
540        * This is expected to be temporary so just retry.
541        */
542       if (errno == EBUSY)
543         continue;
544       else
545         break;
546     }
547
548     prevcpu = vcpu;
549     rc = (*handler[vmexit[vcpu].exitcode])(ctx, &vmexit[vcpu],
550                &vcpu);
551     switch (rc) {
552     case VMEXIT_SWITCH:
553       assert(guest_vcpu_mux);
554       if (vcpu == -1) {
555         stats.cpu_switch_rotate++;
556         vcpu = fbsdrun_get_next_cpu(prevcpu);
557       } else {
558         stats.cpu_switch_direct++;
559       }
560       /* fall through */
561     case VMEXIT_CONTINUE:
```

⑦ 调用设备模拟器（→ 322）

⑯ in命令的时候，把读入的结果存入客户机的rax寄存器。这次解说的是out命令，不执行这一分支的程序（→ 323）

⑰ 返回VMEXIT_CONTINUE（→ 327）

⑤ 由于VMExit的EXITCODE是VM_EXITCODE_INOUT，调用vmexit_inout()（→ 509）

⑲ 在while循环中再次执行vm_run()，客户机运行再度开始（→ 532）

③ 由ioctl返回到这里（→ 533）

④ 调用与EXITCODE相应的处理。这次按in/out命令引起的VMExit进行解说（→ 550）

```
562     rip = vmexit[vcpu].rip + vmexit[vcpu].inst_length;   ← ⑱客户机的rip跳到下一条命令
563     break;
564   case VMEXIT_RESTART:
565     rip = vmexit[vcpu].rip;
566     break;
567   case VMEXIT_RESET:
568     exit(0);
569   default:
570     exit(1);
571   }
572 }
573 fprintf(stderr, "vm_run error %d, errno %d\n", error, errno);
574 }
……(省略)……
757: }
```

Inout.c

Inout.c是模拟I/O命令的程序。实际上担当着管理各个I/O端口设备模拟器处理程序的工作。接到处理请求后，调用设备模拟器处理程序。被调用的处理程序进行实际的模拟处理。

▼ 代码清单3　usr.sbin/bhyve/inout.c

```
……(省略)……
72 int
73 emulate_inout(struct vmctx *ctx, int vcpu, int in, int port, int bytes,
74      uint32_t *eax, int strict)
75 {
76    int flags;
77    uint32_t mask;
78    inout_func_t handler;
79    void *arg;
80
81    assert(port < MAX_IOPORTS);
82
83    handler = inout_handlers[port].handler;   ← ⑧取出各个端口号注册的I/O处理程序
84
85    if (strict && handler == default_inout)
86      return (-1);
87
88    if (!in) {
89      switch (bytes) {
90      case 1:
91        mask = 0xff;
92        break;
93      case 2:
94        mask = 0xffff;
95        break;
96      default:
97        mask = 0xffffffff;
98        break;
99      }
100     *eax = *eax & mask;
101 }
102
```

```
103   flags = inout_handlers[port].flags;
104   arg = inout_handlers[port].arg;
105
106   if ((in && (flags & IOPORT_F_IN)) || (!in && (flags & IOPORT_F_OUT)) )
107     return ((*handler)(ctx, vcpu, in, port, bytes, eax, arg));
108   else
109     return (-1);
110 }
……(省略)……
141 int
142 register_inout(struct inout_port *iop)
143 {
144   assert(iop->port < MAX_IOPORTS);
145   inout_handlers[iop->port].name = iop->name;
146   inout_handlers[iop->port].flags = iop->flags;
147   inout_handlers[iop->port].handler = iop->handler;
148   inout_handlers[iop->port].arg = iop->arg;
149
150   return (0);
151 }
```

⑨取出各个端口号注册的处理程序

⑩由register_inout()注册I/O端口处理程序

consport.c

consport.c是BHyVe专用的半虚拟化控制台驱动程序。引入了UART（Universal Asynchronous Receiver Transmitter）模拟器后，该驱动程序不再是必需程序了。但作为设备驱动程序，它的结构最简单，我们用它来解说设备驱动程序。

▼ 代码清单4　usr.sbin/bhyve/consport.c

```
……(省略)……
95 static void
96 ttywrite(unsigned char wb)
97 {
98 (void) write(STDOUT_FILENO, &wb, 1);
99 }
100
101 static int
102 console_handler(struct vmctx *ctx, int vcpu, int in, int port, int bytes,
103     uint32_t *eax, void *arg)
104 {
105   static int opened;
106
107   if (bytes == 2 && in) {
108     *eax = BVM_CONS_SIG;
109     return (0);
110   }
111
112   if (bytes != 4)
113     return (-1);
114
115   if (!opened) {
116     ttyopen();
117     opened = 1;
118   }
119
120   if (in)
```

⑮ttywrite()用write()向标准输出设备写字符

⑮在console_handler()里，如果I/O的方向是in则执行ttyread()，如果I/O的方向是out则执行ttywrite()，对标准输入输出设备进行I/O处理

```
121     *eax = ttyread();
122   else
123     ttywrite(*eax);
124
125   return (0);
126 }
127
128 static struct inout_port consport = {
129   "bvmcons",
130   BVM_CONSOLE_PORT,
131   IOPORT_F_INOUT,
132   console_handler
133 };
134
135 void
136 init_bvmcons(void)
137 {
138
139   register_inout(&consport);
140 }
```

⑭ 这次解说的是执行out的情形，eax里指定的写操作值被传递给ttywrite()（第123行）

⑫ 把注册的处理程序指定为console_handler()（第132行）

⑪ 启动consport时，在这里注册处理程序（第139行）

小结

本期我们解说了接收到由I/O命令引起的VMExit后，在用户区进行模拟处理的源程序。

现在，有关超级系统管理程序执行循环的一系列源程序，我们就全都讲解完了。下一期我们讲解virtio的原理。

Universal shell programming研究所　http://www.usp-lab.com
上田隆一(UEDA Ryuichi)　Twitter @uecinfo　译/芳龙

第19回 编写CGI脚本(1)——用标准输出向Web服务器传输数据

使用shell脚本可以编写CGI脚本吗?

从本期开始，我们来谈一谈笔者最初便想言及的CGI脚本编写问题。

CGI是通用网关接口(Common Gateway Interface)的缩写，简而言之就是从浏览器启动Web服务器端程序的一种技术。CGI这个词实际上指的是接口(Interface)，所以通过CGI执行的程序，称为CGI程序或者CGI脚本，都是比较贴切的。不过用脚本语言编写时，叫做CGI脚本更好些。

CGI程序可以用任何语言编写。虽说用C语言编写也可以，但这个领域中多使用轻量级语言(Lightweight Language)进行编写。以前使用Perl、PHP的多一些，最近也经常用Ruby、Python编写。

上述语言再加上shell脚本是本章及以后几章我们将探讨的内容。经常有人问：shell脚本可以很容易地使用系统命令，安全性没问题吧?确实，shell脚本的功能远不只实现Web数据交互这样的目的。而且简短的命令也能实现一些邪恶之事，比如`rm -Rf /`(只有8个字符!)等。

关于“是否容易遭受注入攻击”这个问题，笔者认为，虽然在语言角度上它和别的语言没什么大的差别，但小心使用的话，shell脚本反而不容易遭受攻击。仅因为现在市面上的编写手册还不成熟，就武断地认为用shell脚本编写的CGI是很危险的，这是不了解shell脚本的表现。看问题是不应当带有成见。

> 总是谈论女人的人，实际上并不太了解女人；总是诋毁女人的人，实际上完全不了解女人。
>
> ——莫里斯·勒布朗

请把这个格言使用命令sed `'s/女人/shell脚本/g'`执行一下，读一遍，然后再看接下来的内容。

准备Apache

这次，假定的环境是可以运行bash、Apache的UNIX系列操作系统。笔者为了方便，所以使用手边的Mac机。关于Linux系统上运行的介绍，网上可以查到很多，这里就不详细说明了。

笔者也是撰写本文时才知道，OS X系统自带Apache，不用安装立即就可以使用。如图1所示敲入命令，Apache就能启动了。

本连载的读者，可不用浏览器，而是像图2中用curl来确认运行情况。curl可以通过选项指定的URL，把获取到的HTML进行标准输出。

curl一般通过Linux发布的安装包管理工具就可以安装。wget、fetch也有类似的工具。

如图3所示，我们来确认一下CGI的存放目录。CGI程序多存放在cgi-bin目录下，在cgi-bin下，先搜索一下配置文件（httpd.conf）。

打开编辑器，就可以使用编辑器的功能进行查找。但是，这样的说明步骤，不如shell式的描述更容易理解。刚才说的curl也一样，比使用浏览器的截图更轻松了许多。当然在交互上花费的时间还是有差异的。虽为题外话，但这几点实际上是很重要的。

言归正传，从查找结果可以看到，CGI在/Library/WebServer/CGI-Executables/下，是个比较繁琐的目录。遗憾的是，我们就要把CGI脚本放到这里。不过要全部记住这个目录实在太麻烦了，所以我们如图4那样，在自己的根目录下做一个链接。反正只有自己用，顺便也把所有者改一下。

CGI程序只是程序而已

从现在开始进入正题。首先要做的就是运行CGI程序。有很多人一听到讲CGI程序就觉得是很特别的东西，实际它也没什么特别之外。接下来就给大家解开这个疑惑，先做一个简单的尝试。

首先，在目录/tmp/下建立一个名为hoge的文件。把所有者改为可执行Apache的用户。可执行Apache的用户或组通过如下方法来查找。

```
$ grep ^User /private/etc/apache2/httpd.conf
User _www
$ grep ^Group /private/etc/apache2/httpd.conf
Group _www
```

找到用户和组以后，用如下方法更改hoge所有者。

```
$ touch /tmp/hoge
$ sudo chown _www:_www /tmp/hoge
```

然后把rm命令复制到cgi-bin目录下，后缀是.cgi。

```
$ cp /bin/rm ~/cgi-bin/rm.cgi
```

用浏览器执行一下rm.cgi。此时如果使用curl的话看不到效果，所以选择使用浏览器来执行。在地址栏中输入http://localhost/cgi-bin/

▼ 图1　启动Apache

```
$ sudo -s
# apachectl start
# ps cax | grep httpd
16023   ??  Ss     0:00.15 httpd
16024   ??  S      0:00.00 httpd
```

▼ 图2　使用curl进行确认

```
$ curl http://localhost
<html><body><h1>It works!</h1></body></html>
```

▼ 图3　查找cgi-bin目录

```
$ apachectl -V | grep conf
 -D SERVER_CONFIG_FILE="/private/etc/apache2/httpd.conf"

$ cat /private/etc/apache2/httpd.conf | grep cgi-bin
    ScriptAliasMatch ^/cgi-bin/((?i:webobjects)).*$) "/Library/WebServer/CGI-Executables/$1"
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
```

▼ 图4　建立根目录上的快捷链接

```
$ ln -s /Library/WebServer/CGI-Executables/ ./cgi-bin
$ cd cgi-bin
$ sudo chown ueda:wheel ./
```

rm.cgi?/tmp/hoge。

不过很遗憾，浏览器显示的是如图5所示的服务器内部错误（Internal Server Error）。

而且，如下所示，/tmp/hoge无法找到。

```
$ ls /tmp/hoge
ls: /tmp/hoge: No such file or directory
```

你是不是吃了一惊？

到底发生了什么呢？通过浏览器访问http://localhost/cgi-bin/rm.cgi?/tmp/hoge，就相当于把/tmp/hoge当做选项参数传给服务器（本文中指我的Mac）cgi-bin目录下的rm.cgi，从而将/tmp/hoge删除了。使用ssh对远程服务器执行以下命令是不会发生任何变化的：

```
$ ssh  '~/cgi-bin/rm.cgi /tmp/hoge'
```

其不同也就是不使用22号端口，而是80号端口。不过，即使rm命令可能会被互联网

▼ 图5　rm.cgi的执行结果

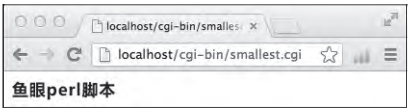


▼ 图6　最简单的CGI脚本

```
$ cat smallest.cgi
#!/bin/bash -xv

echo "Content-Type: text/html"
echo ""
echo 鱼眼perl脚本
写完脚本后设置为可执行
$ chmod +x smallest.cgi
```

▼ 图7　用浏览器执行smallest.cgi的结果



上大量来路不明的人执行，也没什么问题，这是因为Apache有以下限定：

- 设置用户或组来限定可执行用户
- 只执行特定目录下的程序
- 限定只有已经登录的后缀才能执行

这样的话，通过设置一定的制约条件来尽可能确保安全。

反之，如果把~/cgi-bin下的程序都设置成可执行的，那么不管是什么程序都可以由CGI启动。比如rm.cgi这种由C语言编写的脚本，或者传统的Perl编写的脚本都可以运行。也就是说，shell脚本也可以运行。

编写CGI shell脚本

下面，我们尝试用shell编写CGI脚本。首先，如图6所示，这是最简单的CGI脚本，可以在浏览器中显示文字。这个shell脚本没有什么特别之处，在终端上正常执行即可。

```
$ ./smallest.cgi 2> /dev/null
Content-Type: text/html

鱼眼perl脚本
```

在浏览器上运行的话，结果如图7所示。这个例子有几点需要注意。首先，`Content-Type: text/html`是HTTP协议规定的HTTP首部。刚才在浏览器中执行rm.cgi出错就是因为缺少HTTP首部。浏览器和Apache是通过HTTP协议进行交互的，Apache（正在运行的CGI程序）没有返回HTTP首部，所以浏览器就出错了。

HTTP首部下面的echo ""是为了区分头和主体而设置的空行。在头的前面不能加其他东西，比如图8中的脚本，用浏览器执行时是会报错的。

图6的`Content-Type: text/html`意指发送文本形式的HTML。虽然实际上只传输了一行文本，不过现在不需要在这里太纠结。

接下来应该注意的是，shell脚本只是把文字进行了标准输出，并没有对浏览器和Web服务器做什么特别的事情。这是因为Apache会获取shell脚本的输出并传给浏览器。shell脚本应该做的就是保证输出正确的HTTP首部。这也是很典型的UNIX动作。

#行（#!/bin/bash）带有输出日志的选项-xv，不过日志输出到哪里了呢？实际上是如图9所示，被输出到Apache的错误日志里了。

到此为止，所列举的shell要点，与其他轻量级语言并无不同。区别点在于，轻量级语言有便于使用的库，可以很好地封装与Web服务器的直接交互。但是，不管是用什么语言编写，一般的CGI最终都是通过HTTP输出HTML和JavaScript。

编写用于输出的CGI脚本

向浏览器传输文本并让其处理

既然已经明白了可以用shell脚本编写CGI脚本，那么我们马上来写一个试试。实用类型的例子我们放在以后的章节里讲解，这次先来做一个有趣的东西。

首先，被浏览器调用时，编写有如下动作的CGI脚本。

- 让浏览器（客户端）等待
- 在服务器端，一旦把字符串传给CGI脚本，就会在浏览器显示这些字符

CGI脚本的执行并不是一闪而过，而是途中会有暂停。这并没有什么特殊功用，而是为了能够真切地感觉到它的执行。那么，我们编

▼ 图8　HTTP首部前面有东西时会报错

```
$ cat dame.cgi
#!/bin/bash -xv

echo huh?
echo "Content-Type: text/html"
echo ""
echo 湾岸python script
$ curl http://localhost/cgi-bin/dame.cgi 2> /dev/null | head -n 3
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
```

▼ 图9　CGI脚本的标准错误输出都保存在error_log文件里

```
$ cat /private/var/log/apache2/error_log
...
[Tue Apr 23 21:46:14 2013] [error] [client ::1] #!/bin/bash -xv
[Tue Apr 23 21:46:14 2013] [error] [client ::1]
[Tue Apr 23 21:46:14 2013] [error] [client ::1] echo "Content-Type: text/html"
[Tue Apr 23 21:46:14 2013] [error] [client ::1] +echo 'Content-Type: text/html'
[Tue Apr 23 21:46:14 2013] [error] [client ::1] echo ""
[Tue Apr 23 21:46:14 2013] [error] [client ::1] +echo ''
[Tue Apr 23 21:46:14 2013] [error] [client ::1] echo ??? ...
```

写一个如代码清单1的脚本。

第3行的mkfifo命令是用来创建被称作“命名管道”特殊文件的命令。“命名管道”正如其名，就像一条管道，从一头输入字符，另一头就可以输出这些字符。例如：

```
$ echo hoge | cat
```

上述处理若用命名管道编写，就会变成如下的形式：

```
//终端1
$ cat /tmp/pipe
//终端2
$ echo hoge > /tmp/pipe
```

这样，终端1的cat命令在/tmp/pipe传来字符之前，会处于等待状态，直到终端2的echo hoge被执行后，才输出hoge。echo hoge执行结束后，cat也随之结束。仔细一想，这样的动作和普通的管道是一样的。不过，/tmp/pipe不用rm删除的话就会一直存在。

第4行的chmod是权限变更，可以让/tmp/pipe所有者以外的人也写入字符。

接下来，我们尝试用浏览器调用一下nofity.cgi。CGI脚本在cat /tmp/pipe处停下来，所以浏览器也处于等待状态。

此时，在终端输入图10的内容[①]。如果弹出图11那样的警告信息就说明成功了。这个例子虽然没有什么实际用途，但是很有趣。

顺便说一下，为了证明HTTP首部是有意义的，可以把notify.cgi改写成代码清单2（把Content-Type换成text/plain），再执行一遍试试看。

这次，笔者认为如果是正规的浏览器，浏览器上会显示<script>alert(“no more XSS”)</script>这样的字符串。

▼ 代码清单1　notify.cgi

```
01 #!/bin/bash
02
03 mkfifo /tmp/pipe
04 chmod a+w /tmp/pipe
05
06 echo "Content-Type: text/html"
07 echo ""
08 cat /tmp/pipe
09 rm /tmp/pipe
```

▼ 代码清单2　notify.cgi

```
01 #!bin/bash
02
03 mkfifo /tmp/pipe
04 chmod a+w /tmp/pipe
05
06 echo "Content-Type: text/plain"
07 echo ""
08 cat /tmp/pipe
09 rm /tmp/pipe
```

▼ 图10　输入的字符串

```
$ echo '<script>alert("no more XSS!!")</script>' > /tmp/pipe
```

▼ 图11　浏览器弹出警告信息



下载文件

既然已经谈及HTTP首部，那么我们最后来谈一谈文件的下载。比如，可以执行下载大家都喜欢的Excel文件的CGI脚本，如代码清单3所示。

第6行application/octet-stream是指将以二进制形式传输。第7行是指保存为名为hoge.xlsx的文件。第8行是指输出变量LENGTH所指定的长度。

实际上，发送文件发到浏览器的是，第10行中我们所熟悉的cat命令。cat命令不区分文本和二进制，如果进行区分，就不能和其他命令一起使用了。

① 这里需要注意，如果不写/tmp/pipe的话，就会生成一个名为/tmp/pipe的普通的文件。

文件存在的话，所有的文件都会被下载下来，但是HTTP首部要相应做一些微调。比如，如果想用浏览器直接观看mpeg文件，就需要编写代码清单4中的脚本。

以笔者平时使用的浏览器（Mac的Google Chrome和Firefox）为例，脚本会启动图12那样的浏览器插件，播放动画。

若在头上加上Content-Disposition: attachment; filename="hoge.mpeg"的话，就会被询问是否把文件作为可再生文件保存起来，此时可以不进行播放，直接将文件保存起来。

关于HTTP首部的知识，笔者所知大致如此。如果在其他语言中，间接地操作过HTTP首部，也应该能很好地控制shell脚本吧。

总　结

本章主要介绍了使用shell脚本编写CGI脚本的方法，尤其是关于输出方面。本章最重要的内容是，经由Apache使用标准输出向浏览器传输内容。这些内容，也是互联网与UNIX共同发展的痕迹。与其说是痕迹，不如说是必然的发展。标准输入输出，没有比这更抽象的接口了，对此，首先应该探讨的是其使用方法。

下一章，我们将用CGI脚本做些东西，并使用到POST、GET设置。

▼ 代码清单3　执行下载文件的CGI脚本

```
01 #!/bin/bash -xv
02
03 FILE=/tmp/book1.xlsx
04 LENGTH=$(wc -c $FILE | awk '{print $1}')
05
06 echo "Content-Type: application/octet-stream"
07 echo 'Content-Disposition: attachment; filename="hoge.xlsx"'
08 echo "Content-Length: $LENGTH"
09 echo
10 cat $FILE
```

▼ 代码清单4　观看mpeg文件的CGI脚本（download_movie.cgi）

```
01 #!bin/bash
02
03 mkfifo /tmp/pipe
04 chmod a+w /tmp/pipe
05
06 echo "Content-Type: text/plain"
07 echo ""
08 cat /tmp/pipe
09 rm /tmp/pipe
```



图12　正确书写HTTP首部的话浏览器就会有适当的动作▶

可以使用SDN吗？应该使用SDN吗？

网络虚拟化的陷阱

Data Hotel株式会社
伊势幸一（ise kouichi）
Twitter @ibucho
译/苏祎

第2回

端点模型的验证
——VXLAN、NVGRE、STT、独自扩展问题

从Fabric到端点模型

以前，云平台上的网络虚拟化主要有两个目的。一是很多租户（tanent）在一个网络系统上运行时，可以生成超过基于IEEE802.1Q的最大VLAN个数（4094）的租户段。二是形成一个多租户虚拟机（后称VM：Virtual Machine）下，无需变更IP地址和MAC地址，就能在IaaS平台上自由移动、启动、停止的网络。

上回提到使用Fabric模型实现网络虚拟化，这种方法在数据链路层提供了复用和虚拟段，但不支持网络层以上的虚拟化。而端点模型（Endpoint Model）则能虚拟化网络层以上的层级，解决VLAN的上限问题，从而形成灵活的网络段。

端点模型技术在活用现存物理网络，又不依赖于物理拓扑的条件下，即可形成了逻辑上的网络段。大多数情况下，指在现有的IP网络上，使用隧道技术，叠加以太网L2段。使用隧道技术，不仅可以形成叠加网，还可以重新定义隧道首部中识别租户段的ID，使生成含有4094以上个租户段成为可能。

一般来说，隧道的端点有两种形式：物理机上的宿主管理器（hypervisor）和物理交换机（图1）。把物理交换机作为隧道的端点来使用的话，会在投入产出比上出现问题，所以，大多情况下仅作为物理网络和虚拟网络之间的网关（gateway）来使用。还有另一种端点模型，即把虚拟交换机的工作交给网卡（NIC）或相邻交换机中协助处理任务的VEB（Virtual Ethernet Bridge）和VEPA（Virtual Ethernet Port Aggregator），但因其不能扩展VLAN个数的上限，也不能形成L2叠加，所以只能在这里被舍弃掉了。

现在具有代表性的端点模型协议有：VXLAN（Virtual eXtensible Local Area Network）、NVGRE（Network Virtualization using Generic Routing Encapsulation）、STT（Stateless Transport Tunneling）。和上回一样，

我们来讨论一下这3种协议的概要、缺点及致命的缺陷。

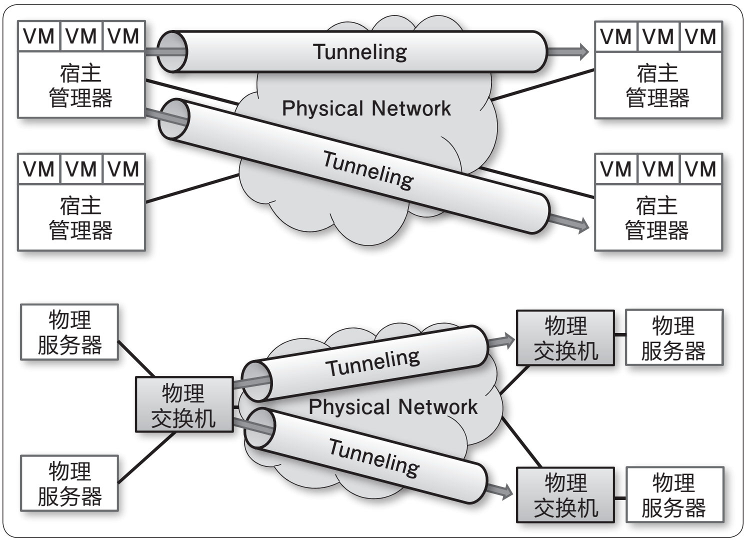
VXLAN

VXLAN是由VMware、Cisco System、Alista、Broadcom、Citrix Systems和Red Hat这6家公司提出的虚拟网络协议。第一份IETF草案于2011年8月公布。如图2所示，VXLAN把原始的Ethernet帧封装到UDP包中，使用现有的IP网进行传输，同时，使用VXLAN专有头中24位长的段ID（VNI：VXLAN Network ID），从理论上把VLAN个数提高到1600万以上。

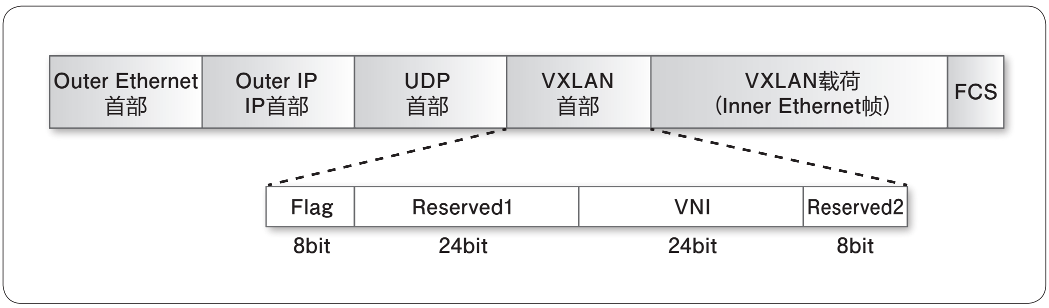
VXLAN封装功能使用了内核UDP服务，既可在内核组装，又可在用户应用程序里实现。从性能上考虑，在内核里实现会更优，但由于对UDP服务的消耗几乎可忽略不计，从实现方便和开发省时的层面考虑，在用户应用程序里实现也没有问题。

事实上，VXLAN协议的首部只含有标志位和VNI，以隧道处理的消耗也非常低。并且，虽然VXLAN首部是新定义的格式，但因为是在UDP载荷里，所以在互联网协议栈上来看，可以被看做是应用数据。因此，VXLAN帧对路径上经由网络设备的转发没有影响，物理服务器的网卡、现有的物理网络设备、NAT和防火墙等网络组成元素都可以照常使用。

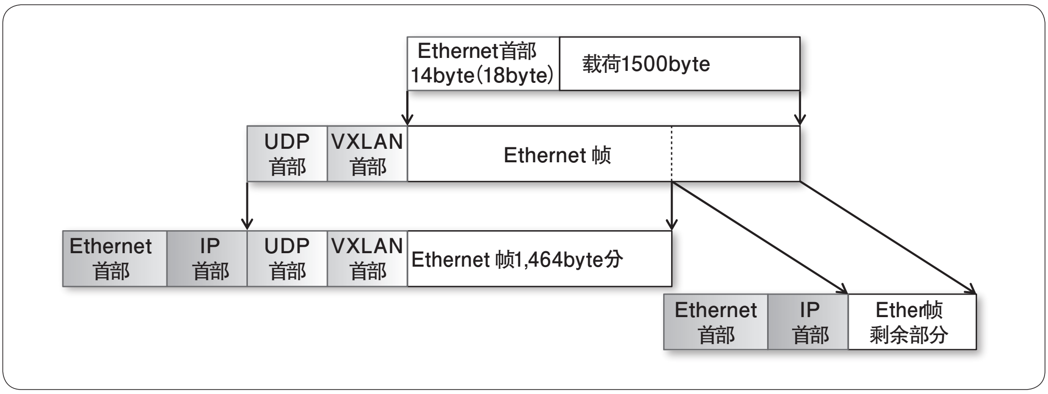
▼ 图1　端点模型



▼ 图2　VXLAN帧与头部



▼ 图3 使用IP分片进行传输



MTU的问题

隧道协议中，将UDP与VXLAN加入到传输中的Ethernet帧载荷里时，若原始Ethernet帧大小为1500字节，就无法共存于一个Ethernet载荷里。所以长1500字节的原始Ethernet帧就要被分割成多个Ethernet帧来传输（图3）。

关于复原原始帧，通过IP层面的组装，就能一定程度上保证UDP层面的校验。但若云平台上传输的帧，几乎都是载荷1500字节，这可能对网络性能有显著影响。网络设备端口性能一般用发送接收的数据量来衡量，转发性能依赖于包的数量。换句话说，机器内部处理转发，即是处理各个协议层的首部，并决定转发地址接口。若原始帧经常被分割为两份进行传输，简单来说，就会导致机器的转发性能被降低了一半。

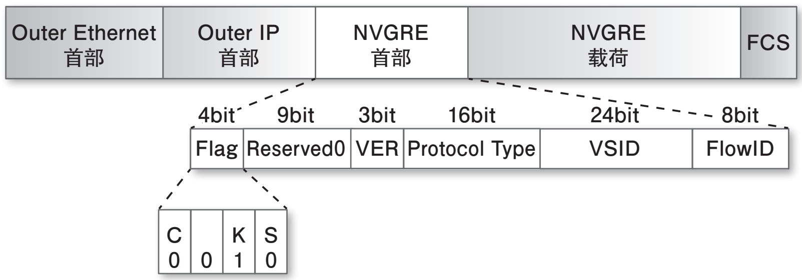
VXLAN供应商针对IP分片化问题，推荐如下处理：增大物理网络的MTU，使VM能识别的MTU上限为1600字节以上；或设置各个VM的MTU为1400字节。虽然有支持提高Ethernet MTU大小的设备，但期待所有的交换机和路由器都支持设置是不现实的。

在物理网络上增大支持的MTU，会一定程度上造成数据链路协议兼容性和设备投资成本上的困难。比较现实的方案是在VM上缩小MTU。这种情况下，可以把所有虚拟机OS镜像，设定成新的MTU大小，或者使用Chef管理工具在搭建VM时调整MTU。不管使用哪个方法，为了将运行适配稳定，需要花费很多的工夫和成本。

NVGRE

NVGRE 是 Microsoft、Alista、Intel、Dell、HP、Broadcom和Emulex这7家公司提出的协议。差不多与VXLAN同时期在IETF草案中公布。NVGRE是一项虚拟网络实现技术，它基于对RFC2784和RFC2890下标准化GRE（Generic Routing Encapsultation）协议

▼ 图4 NVGRE帧与头部



key option域的扩大解释。NVGRE把VM发送的Ethernet帧和NVGRE首部、传输用IP首部一起封装，并通过物理网络进行传输。

如前所述，NVGRE与GRE的首部格式完全相同。首部的前4位标志位从左到右为校验位（checksum bit）、未使用、key位（key bit）和序列位（sequence bit）用来表示选项的设置与否。NVGRE中只有key位设置为1，其他都设置为0（图4）。

32位的key option域里，前24位被用来代表标示NVGRE段的VSID（Virtual Subnet ID）。这相当于VXLAN里的VNI。后面8位定义了物理网络中ECMP的flow id。NVGRE只在key option的解释上和GRE不同，其他的诸如格式、协议号和GRE一模一样。因此，对于网络设备和OS来说，GRE包和NVGRE包没有任何区别。所以，现有的网络设备和服务器的网卡、OS的Ethernet和IP驱动都不需要做任何修改。

但是，GRE一般是在OS的内核进行处理，为了支持NVGRE协议，必须修改内核或宿主管理器的源代码，和VXLAN相比，实现和维护的成本大大增加。

校验问题

NVGRE上必然也存在和MTU相关的问题，且与其他协议相比，问题更严重。假定出口IP首部长度为基准的20字节，包含GRE首部的IP载荷超过1481字节时，必然会在发送方产生分片化。接收方会对分割后的IP包进行重新组装，但因为在NVGRE首部不包含校验option，也就没有办法确认复原后的IP载荷的正确性了。

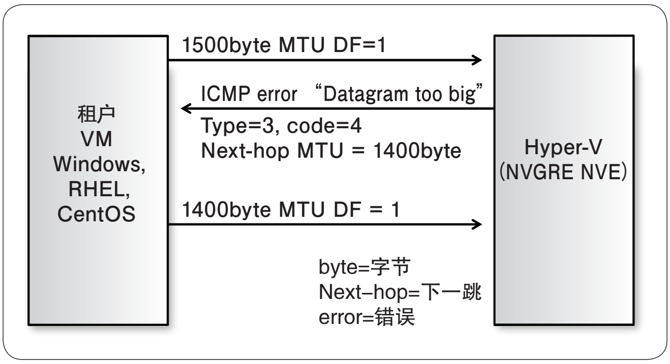
一般在Ethernet帧的最后面，会有一个FCS（帧检验序列，Frame Check Sequence）来检验数据的正确性。这个FCS（大多数情况下）都是通过网卡使用硬件进行处理的，网卡把FCS去除后交给上一层的驱动，或者上一层的驱动把没有FCS的帧交给网卡。因此，在宿主管理器或VM上不能检查使用了FCS原始帧的正确性。这样也就不能保证NVGRE恢复后的正确性，万一在传输过程中帧中混入了杂音被破坏，这个受损的帧就这样交付给VM的话，可能会使VM发生故障。

现在，实现了NVGRE的微软的Hyper-V

里，从VM发送出的帧负载若为1500，会在返回错误消息“ICMP Datagram too big”的同时，通知VM一个可传送的MTU大小（例如1400字节），避免在传输NVGRE包的过程中出现分片化。

和VXLAN一样，对于MTU问题，可以使用工具把所有VM镜像的MTU调整为合适的值，或者统一使用已经实现了NVGRE和MTU调整的微软产品。不管哪一个都会增加使用成本，可能也会形成受制于一家厂商的情况。

▼ 图5　通过宿主管理器进行MTU调整



STT

STT是NVP（网络虚拟平台，Network Virtualization Platform）上实现的独立隧道协议，由被VMware收购的Nicira所开发。2012年1月作为IETF草案公布后，成为了开放协议。STT使用TCP包，把最大64 KB载荷以隧道化的方式，在3层网络上进行传输，从而搭建了一个虚拟的叠加网络。

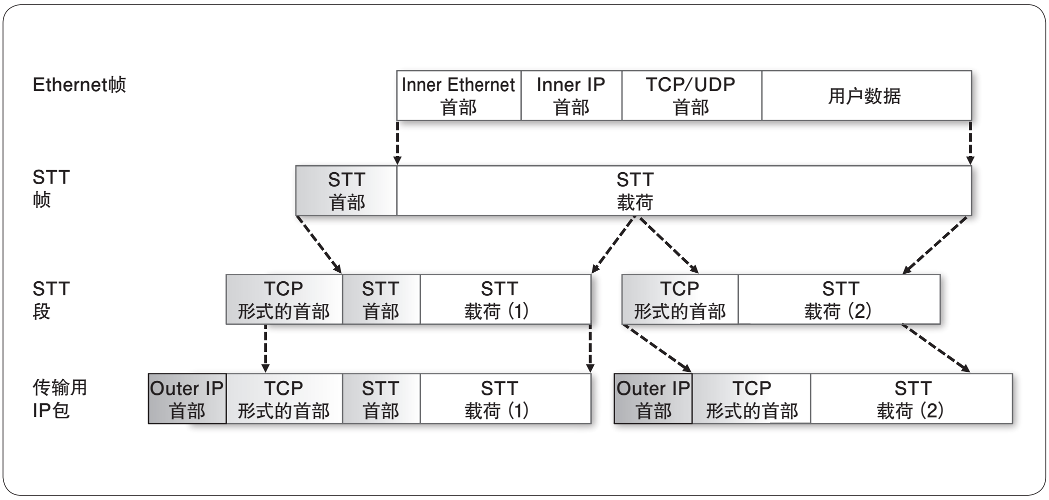
其他的端点模型以原始Ethernet帧作为载荷为前提，Ethernet之外，STT还支持帧中继、ATM这样更大的载荷数据链路帧。但是，云平台和数据中心内基本上都是通过Ethernet来使用网络，64 KB这样的载荷长度基本没有意义。这个最大64 KB的值，是由后面会提到的16位STT段位移（segment offset）值所确定的，并不是通信性能和各个协议的最大MTU大小所决定的。

STT中，宿主管理器把从VM处获得的帧首先用STT首部进行封装，制作成STT帧。接着把STT帧分割成可在物理网络上传输的片段，每个片段都用扩展TCP首部进行封装（图6）。分段后，这些片段称为STT段（segment）。STT段会继续根据物理网络相关的IP包进行封装，并从发送方的物理机器传输到接收方虚拟机所在的物理机器。

与VXLAN、NVGRE相比，STT首部构造更加复杂。如图7所示，含有版本号、标志位、L4偏移量、最大段长、PCP、V位、VLAN ID和上下文ID等信息。64位的上下文ID一般用来作租户段的标识符，它的作用不仅限于租户段，还定义了通用ID，以指示收到STT帧端点时应该进行的处理。虽然和VXLAN的VNI、NVGRE的VSID作用相同，但对虚拟网络段，STT并没有规定以一一对应的形式分配。或者说，可能多个段分配一个ID，可能一个段分配了多个上下文ID，也可能发往同一个租户的帧使用不同的处理方式。

封装STT段的扩展TCP首部格式与通常

▼ 图6　STT封装序列

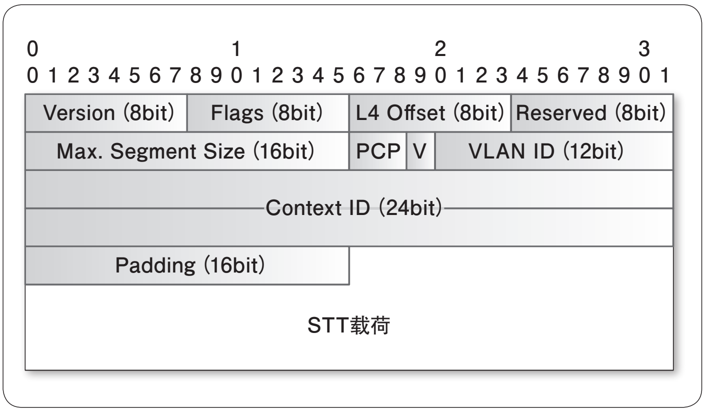


格式相同，只是RFC793定义下的TCP头序列号和响应确认序号域，会被用来表示STT帧长(Frame Length)、STT段偏移(segment offset)和STT帧标识符(Frame Identification)，以执行独立的处理(图8)。STT帧标识符以帧为单位进行分配，STT段按照底层网络能传输的大小进行分割，在STT段被重新组装时，这个标识符可以区分帧的段。

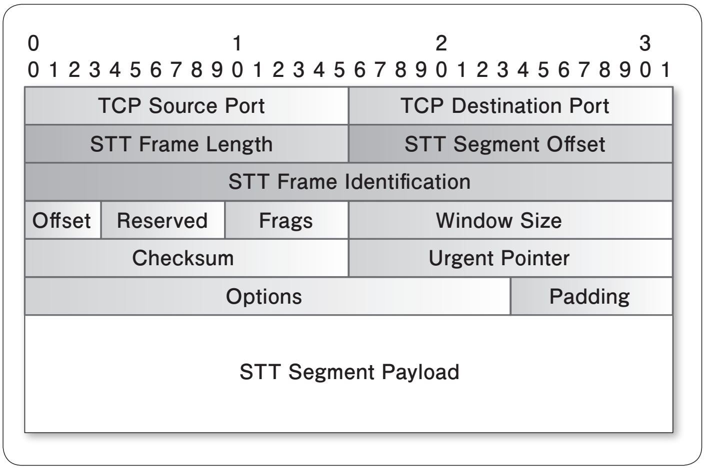
扩展TCP首部中的6位标志位域和RFC793中的定义相同，表示应答序号有效的ACK位和表示将接收数据直接递交给上层应用的PSH位恒常设置为1，其他设置为0。

STT的载荷长度可达

▼ 图7　STT首部格式



▼ 图8　扩展TCP首部格式



64 KB，可以完全被容纳最大长度为1514字节的IEEE802.3帧和最大长度为1518字节的IEEE802.1Q帧，乍看起来好像MTU问题似乎不存在了，STT帧在使用TCP进行封装时，会被分割成底层网络的MSS（最大段长）所限制的大小，底层网络是Ethernet IP的话，最终会在IP层进行分片。因此，与VXLAN、NVGRE一样，STT也不能避免因为帧分片所导致的网络性能降低。

但是，STT存在另一个致命缺陷。VXLAN的隧道处理，事实上只处理含标志位和VNI的VXLAN首部和UDP首部。NVGRE的首部，是在VXLAN的基础上，添加了版本、协议类型、flowid，因为沿用了原来的GRE首部，所以不需要VXLAN的UDP这样的传输层头部，处理也更简便。这使得NVGRE也可以有类似于VXLAN的处理步骤了。

如图7所示，STT首部格式的内容比VXLAN和NVGRE更加复杂，与VXLAN相比，又多了版本、L4偏移量、最大段长、PCP和V位、VLAN ID域，虚拟网络接口需要处理的域是VXLAN的两倍以上。并且STT段使用TCP首部进行封装，TCP首部的处理负荷远不如UDP。

因此，STT草案中，提出了把TCP首部的处理交给网卡来处理，以此来避免性能降低。换句话说，因协议处理负荷很大，所以STT是以使用硬件来解决此缺陷为前提而设计的。

这样又会有新的问题。STT段使用扩展TCP首部进行的封装，并不是RFC793中标准化的完整TCP，只是使用了TCP形式的独立首部。NVGRE中，因为沿用了已标准化的GRE格式，传输路径中翻译GRE首部的节点，在处理通过的帧时，也不会产生矛盾。但是，含有扩展TCP首部的STT段的协议号（Protocol Number）使用的是和普通TCP一样的6号，但并不和标准的TCP兼容，传输路径中的节点在解析TCP首部时，可能会因为首部内容不符合TCP标准，而将它废弃掉。这样，分割后的STT帧不能在接收方重新组装，整个帧就可能无法全部传输到接收方。

STT草案中也意识到了这个问题，它暗示道，以后会把STT段的协议号和TCP的区别开来，分配一个新协议号。但是，如果STT段的协议号和TCP的不一样，就无法使用网卡来处理TCP首部了。STT协议最初的设计，是以通过使用网卡硬件处理TCP首部来避免性能降低为大前提的，引入其他协议号的话，必须承受网络性能降低的后果。否则，就需要所有可能会处理STT的网卡都在硬件层面实现处理STT的功能，想必不是所有的生产厂商都会答应这个条件吧。

并且，把TCP的处理都交给网卡也存在问题。网卡不能像CPU那样精确快速地执行中断的响应和处理。因此，当OS和宿主管理器把协议处理转交给网卡，网卡在处理完成变为ready状态前，可能无法响应中断。因此，CPU、宿主管理器、Guest os等部分的中断和上下文切换（context switch），会和网卡控制器时序（timing）产生差异，可能会导致网络操作堆积。这个问题的传统解决方式有如下选择：有意关闭网卡硬件处理；导入更高效的驱动；替换网卡这样的选项。

但在数千台、数万台设备构建而成的云IaaS平台的所有网卡和宿主管理器、Guest os、应用程序间进行检查是非常耗费精力的。

或者说，使用STT的网络虚拟化，效率上可能比VXLAN和NVGRE更低一些，这个低下的效率所换来的是，只有64位上下文ID才能带来的几乎无限的租户段。

小　结

各个端点模型共同的致命缺陷，存在于VM之间通信开始前的部分。在IP网上构建的叠加网络端点模型里，缺少将Mac地址和IP地址对应起来的地址解析方法。IPv4的ARP使用广播来完成地址解析，叠加网络上基本都是建立P2P隧道，因而有时不能进行广播。所以，在VXLAN、NVGRE、STT的草案中，建议使用多播组来完成解析方式。

在目的为安全隔离各个租户间的流量的虚拟网络上，使用一个多播组来完成所有租户的地址解析，就有点本末倒置了。一般是以租户为单位进行分配多播组。或者，端点模型的导入必须有4096个以上的多播组在运行。想必管理过4000个以上多播组的工程师应该也很少有吧。至少笔者是无法想象。

在已经问世的VMware和微软的实际应用中，已经有不少可以管理IaaS平台全体的Orchestrator和Federator，或者通过SDN控制器来完成IP和MAC的地址解析。

并且，端点模型是在物理网络上使用隧道形成L2链接叠加的网络虚拟化技术，不论使用哪种方式，一般来说在附加的协议头和分片化处理上都会发生通信消耗，导致网络性能降低。因此，端点模型的虚拟化技术是不能提高网络的性能的。

如果性能不仅是从传输速度和传送量来考量，还从能否灵活快速地形成拓扑来评价的话，就不是上面的结论了。

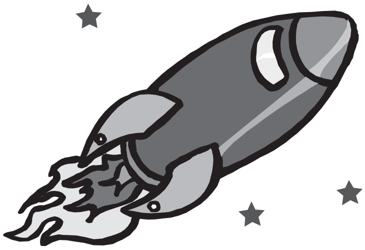
比起服务器间、应用之间的通信速度、数据量，灵活快速地形成租户段，更高效地维护系统整体，应该也是非常实用的技术。

使用端点模型的网络虚拟化更适合如下环境：网络传输速度和吞吐量并没有那么重要，应优先快速添加、删除、修改租户的L2段。

人们很容易陷入一种错觉。即网络虚拟化后，一切问题就都迎刃而解，应该先充分理解和调查各个技术的特点、应用的发展情况，以及环境中自己最看重的因素，然后再来判断是否应该引入这项技术。

文/Debian开发者：山根秀树（henrich@debian.org） 译/马健

5 Debian 7.0 “Wheezy”的变化之处



Debian热点

Debian 7.0 发布!

大家期待已久的Debian 7.0“Wheezy”终于在5月4日顺利发布。其安装要点请参考安装手册①，升级时的注意事项请参考版本注释②。本文将介绍Wheezy的更新内容和背景。

增加了支持的架构

本次Debian的更新依然会增加其支持的架构的种类。主要增加了其对armhf和s390x两种架构的支持。虽然各位读者很少有机会能够接触到s390x这样的大型机架构，但OpenBlocks服务器和目前相当有人气的单片机Raspberry Pi都采用了armhf架构。armhf使用者可以将其与armel进行比较，从而切实感受armhf带来的速度提升。

软件包数量的变化

Wheezy的软件包总数是37 493个，变化如下：

更新：20 160

删除：4125

添加：12 800

这不仅仅有版本升级方面的变化，系统方面也有如下变更。

安全方面的强化

此次更新对一部分的包进行了安全强化（hardening）。即在gcc编译时，将各种各样的安全标志设置为有效，从而缓和外部攻击的威胁。具体的做法是，打包时在源码包的debian/rules文件中显式地设置dpkgbuildflags，或者利用debhelper9之后的版本（debian/compat文件中设置为9）。虽然在Debian开发后期并没有对所有的包进行安全方面的适配，但是，一定会将总体中占比30%左右的用户安装的“base”软件包，曾经出现过安全问题的软件包，以及受到来自网络方面攻击可能性较高的演示包之类的安全标志设置为有效③。另外，在下一个开发周期（代号Jessie）中，将会通过建

① http://www.debian.org/releases/wheezy/installmanual
② http://www.debian.org/releases/wheezy/releasenotes
③ http://outflux.net/debian/hardening/

立这一部分的检查系统[4]，从而大幅提高其覆盖率，这一点也备受期待。

多架构（Multi-Arch）支持

在Wheezy中引入的“单系统安装和运行不同架构的运行库和应用程序机制”便是多架构（Multi-Arch）。例如，在使用当今主流的64bit环境（amd64）系统的同时，能够使用32bit的应用等[5]。利用这一机制，可以导入库，并可以通过内核使用qemu在x86环境中运行ARM[6]。由于在默认状态下并没有进行这样的环境设置，所以读者需要在适当的设置之后才能获得多架构支持。

接下来，我们用Skype（i386）来试验一下吧。在Skype的官方网站上下载其安装包（debian 7.0版）之后，如果想要在amd64框架的机器上使用dpkg -i来安装的话，就会出现图1中的错误。

在这里首先使用

```
$ sudo dpkg --add-architecture i386
```

▼ 图1　在amd64的机器上安装Skype(i386)(其1)

```
$ sudo dpkg -i skype-debian_4.2.0.11-1_i383.deb
dpkg: error processing skype-debian_ ↵
4.1.0.20-1_i386.deb (--install):
软件包的体系架构(i386)与本机系统的架构 ↵
(amd64)不符
在处理时有错误发生:
skype-debian_4.1.0.20-1_i386.deb
```

来指定想要安装的应用程序的框架。

然后在 /etc/apt/sources.list 的

```
deb http://ftp.jp.debian.org/debian/ ↵
wheezy main contrib non-free
```

这一行的位置添加如下代码：

```
deb [arch=amd64,i386] http://ftp.jp. ↵
debian.org/debian/ wheezy main contrib ↵
non-free
```

代码中最重要的是arch=<对象架构1>,<对象架构2>这一部分。添加之后，不要忘记运行apt-get update。全部执行之后，刚刚的错误就会消除，之后便能够进行下一步操作了。这里以管理者身份执行apt-get install -f命令，依存关系的错误就会消失，也就能够安装Skype了（图2）。

内核，X.org

在Wheezy中，Linux内核包名从linux-2.6 package变更为linux package，与此同时，版本也变成了3.2。现在，作为这个版本的维护者Ben Hutchings还负责着上行数据流的维护工作。所以，kernel.org上发布的内核与Debian包中的内核并无多大的区别。而且，世界上最精通内核的专家维护着这个版本的Linux内核程序包。Linux内核包可以说是最为复杂的软件之一，没有人比Ben Hutchings更适合这份工作了。

但是，有些读者可能会问：“现在3.9都出来了，为什么不使用最新版本的内核呢？”这里，要先分析其背景。首先，如果

④ http://buildd.debian.org/~brlink/
⑤ 过去是通过导入ia32-lib package来对应这种问题的。
⑥ http://gihyo.jp/admin/serial/01/ubuntu-recipe/0226?page=2

要发行Debian的稳定版的话，就必须提供至少3年的技术支持，所以，在已发行稳定版的内核中，就必须要选择LTS（Long Term Support）内核。目前，LTS内核的版本有4个，3.4、3.2、3.0和2.6.34，其中3.4.1是2012年6月1日发布的。而Wheezy的开发进入冻结阶段则是在2012年6月30日，大致是在同一时期。这种情况下，很难再改用3.4版本的内核进行开发。另外，这个时期的3.4版本虽然说是LTS，但它还不够成熟。相反，3.2版本则是已经解决和处理了各种bug的成熟版。这样看来，进入冻结阶段的时候，没有理由去选择3.2版本以外的其他版本内核[7]。另外，虽然是3.2版本，新版本的驱动也是会随时移植并导入。Wheezy正式版中不可用的设备也很有可能会在以后发布的测试版和内核更新时变为可用。顺便提一下，Debian不稳定版使用了最新的Linux内核，所以想使用最新内核的读者，可以去尝试一下Debian不稳定版。

▼ 图2　在amd64的机器上安装Skype（i386）(其2)

```
$ sudo dpkg -i skype-debian_4.2.0.11-1_i386.deb
Selecting previously unselected package skype.
(正在读取数据库 ... 系统当前共安装有 132965 个文件⏎
和目录。)
正在解压缩 skype
(从 skype-debian_4.1.0.20-1_i386.deb) ...
dpkg: dependency problems prevent configuation ⏎
of skype:
skype 依赖于 libasound2 (>= 1.0.16)

(中略)

在处理时有错误发生：
 skype

$ sudo apt-get -f install
正在读取软件包列表 ... 完成
正在分析软件包的依赖关系树
正在读取状态信息 ... 完成
正在更正依赖关系 ... 完成
将会安装下列额外的软件包：
  gcc-4.7-base:i386 libasound2:i386 libaudio2:⏎
i386 libavahi-client3:i386

(中略)

解压缩后会消耗掉 101MB 的额外空间。
您希望继续执行吗？ [Y/n]y
获取 : 1 http://ftp.jp.debian.org/debian/wheezy/ ⏎
main gcc-4.7-base i386 4.7.2-5 [143 kB]
    ⋮
```

现在X.org的最新发布版本是7.7版，视频驱动方面都已经适配好了。如果出现冲突，那应该是KMS（Kernel Mode Setting，内核模式设置）和固件之间兼容方面的问题[8]，这种情况下应该将firmware-linux-nonfree等对应的固件进行单独加载，或将启动时的参数设置为radeon.modeset=0，同时，还必须要禁用KMS。

工具链

此次更新，也对重要的工具链进行了正式的版本升级。在编译器方面，虽然没有加入刚刚最新发布的Gcc4.8，但采用了Gcc4.7.2，这在当前，应该足以满足开发者的需求了吧。另外，LLVM（clang）则使

⑦ 笔者也曾经疑惑："为什么不用3.4呢？"得到的回复是"3.2最可靠的用来发布的版本"。
⑧ 笔者用的是带Amd CPU的Thinkpad X121e笔记本，Radeon HD 6250显卡恰恰遇到这个问题，因此我吃了不少的苦头。

用了3.0版本。想要使用最新版本的读者可以利用由LLVM项目提供的LLVM Debian/Ubuntu nightly packages[⑨]。

如果下次Debian 8.0"Jessie"的时候也能够以同样的方式开展活动就好了。

Wheezy发布庆祝活动

关于桌面的解说就放到下期，这里来说一下一些近期的活动吧。为了庆祝Wheezy的发行，世界各地都展开了大大小小的庆祝聚会。日本恰好是黄金周，所以时间上错开了大约一周，5月11日东京涩谷开展了庆祝聚会[⑩]。7月迎来了第100回的东京区Debian学习会，在学习会（照片1）结束后大家还小聚了一下。

▼ 照片1　东京区域Debian学习会情况



▼ 图3　大统一Debian学习会2013活动网站



⑨ http://livm.org/apt/
⑩ 提供会场的是VOYAGE GROUP，在这里对其表示感谢。



红帽与富士通的关联

小崎 资广
KOSAKI Motohiro
富士通Linux软件开发统括部所属
译/王凤波

什么是Onsite工程师

大家好！我是富士通的内核工程师小崎。初次见面，请多关照。

笔者在位于波士顿郊外的红帽公司西福特办公室工作。接下来将由我特别为大家奉献一段来自波士顿的报道。

为什么富士通的工程师却坐在红帽的办公室里？说来话长，10多年前，在由Red Hat Linux向RHEL迁移的黎明期，红帽就与富士通达成协议，在红帽公司内部设置了共同开发部门。此后，虽然成员不断更迭，但是这种共同开发的合作关系却一直延续至今。其开发内容包括很多方面，诸如内核转储功能强化、跟踪功能、CPU和内存热移除的实现、通过大规模硬件实现性能强化等。富士通在东京证券交易所或者银行等关键任务开发上拥有非常丰富的经验，非常擅长“可靠性强化”等相关领域。

为什么红帽没有位于西海岸而是位于波士顿呢？我也经常被问到这个问题。红帽原本就是位于东海岸的企业，即便现在，其主要开发基地依然位于东海岸。但搬往西海岸的说法变得越来越明朗，并且就在两年多前，关于是否将办公地点迁移至硅谷一事，就曾在当地引起轩然大波。最后，西福特当局做出妥协，提出将固定资产税减免50%的特别减税政策，红帽才最终决定留下来。这一新闻曾轰动一时。

距离西福特办公室10分钟左右车程，有一个叫纳什欧巴（Nashoba）的滑雪场，驾车再走远一点，还有好几个可以当日往返的像样的滑雪场，对于冬季体育运动爱好者而言，那是个让人很享受的地方。LKML（Linux Kernel Mailing List）的维护人员多受雇于西海岸的企业，所以有一句玩笑说“真正的OS开发者在西海岸的太浩（Tahoe，从西海岸可以当日往返的滑雪场，很多人知道它是因为4.3BSD-Tahoe由此得名）享受滑雪的快乐”，其实位于东海岸的波士顿也着实是个不错的地方。

日常的开发形态

关于内核开发，因为其开发成员大多是上游社区（Upstream Community）的成员，所以其开发氛围与LKML惊人地相似。因为红帽是Linux企业，所以也有人认为这是理所当然的。大家各自将补丁发送至邮件列表，其他开发人员则进行审核并回信反馈审核结果。我推测这种LKML式文化是在前任RH内核维护艾伦·考克斯（Alan Cox）的影响下形成的。不过这个推测未经证实，因为那个人可不是个普通人，当然，这是从积极的意义上来讲。

正如惠比寿第3回连载报道中介绍的那样，RHEL有一个叫上游优先（upstream first）的策略，原则上只有与上游内核合并（merge）的补丁才会被合并。那么，是不是可以说只进行回

迁（backport）会比修改上游内核容易呢？不是的，其实回迁是很复杂的。因为所基于的内核版本不同，有时会导致无法顺利回迁。也有的补丁因为周边代码不同，如果进行回迁，反而会产生新的bug（即所谓的enbug）。有一次，我们实在没有办法而不得不研究其他修改方法时，一位上游内核的维护人员跳出来厉声责问："你为什么不在上游修改？"真是被他一招击倒。然而，对于一个训练有素的开发者而言，其实这是一种赏赐。

上游内核的最新技术是kABI[①]因为兼容失败而不得不仅将补丁的一部分进行回迁时（特别是上游改动规模较大时）是非常棘手的，所以有时候就会有人抱怨说"所以我才在上游进行讨论时特别强调了这个补丁"。最后，还是由负责上游修改的当事人出场，才好歹使问题得到了解决。

会议上的问候

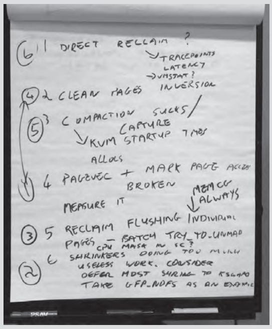
红帽是一个跨国企业，所以即使是通过公司内部邮件经常联系的人，有时也会因为办公地点不同[②]而从未谋面。在一次会议上，我发现有两个人正聊得热火朝天，向他们打听了一下才知道，原来这两个人同属一个开发团队，但却是第一次见面，所以才聊得如此起劲。Linux有很多会议都是在全世界范围内召开的，所以大家会利用这样的机会增进沟通与交流。笔者也曾在上个月于旧金山召开的Linux Storage Filesystem & MM Summit会议上与数位开发人员进行了交流。

① kABI（Kernel Application Binary Interface），内核接口的二进制兼容性，如果这个发生错误，第三方驱动将停止工作。
② 本来就有很多人在自己家里工作。

东海岸的美食

提到波士顿的美食，当然非龙虾莫属。这是各种旅游指南中绝对可以找得到的美味，但令人奇怪的是，当地人并不经常食用龙虾。仔细想来也是，日本三重县的人也不是每天都会吃伊势大龙虾，这是一样的道理。笔者基本上经常换着口味地吃一些汉堡、三明治、墨西哥玉米煎饼、泰国面等。至于日本料理，因为大多是中国人或者韩国人在经营，所以基本上不去。红帽公司员工食堂的汉堡使用的肉是当场煎烤的，所以吃起来口感鲜美，不愧是美国的国民料理啊。由于篇幅所限，这里就不再给大家一一介绍了。如果通过本篇报道，能或多或少地增进了大家对红帽公司开发氛围的了解，笔者将不胜荣幸。

照片1是MM Summit会议上由大家各自提出的、讨论得最白热化的Linux VM相关课题的总结列表，以及该课题的解决先后顺序。
例如：先解决pagevec结构；直接回收（direct reclaim）的延迟虽然很重要，但因为数据较少，所以先增加追踪点；因为回收的延迟，在已经映射（mmap）过页面的处理过程中，不允许try-to-unmap()函数在每个页面都调用IPI而导致发生无效（invalidate）的情况；批处理性等。这些都是大家各自提出并一起整理出来的。用○圈住的数字可以理解成优先度或难易度，表示解决的先后顺序。



照片1　Linux VM的课题列表

第39回 Ubuntu月报

编译Chromium OS

Ubuntu日本团队
Awashiro Ikuya　ikuya@fruitsbasket.info
译/kaku

本文将介绍Ubuntu 12.04推荐编译环境Chromium OS的编译方法和简单的使用方法。

为什么选择Chromium OS

参考上月刊的文章，大家有可能已经在使用Ubuntu 13.04了。除技术支持的时间缩短了以外，笔者认为13.04是一个非常好的版本，本文也是在13.04下执笔完成的。但是本次使用的是Ubuntu 12.04 LTS，所以如果没有此版本的话，必须事先准备下环境。这个要在开始前就完成。

Chromium OS是Google Chrome OS的开放源代码版本，由于日本还没有预装Chrome OS的PC发售，所以只能手动编译装载Chromium OS的PC。当然，如果有非常好的日语支持的话就太好了，但目前我认为在这方面还有待改善。没有预装Chrome OS的PC发售真是很可惜。

编译Chromium OS有如下意义。首先Ubuntu和想象的非常不同，通常只要稍微上手接触一下便会觉得非常有趣。Ubuntu推荐的编译环境，也正是本文所采用的。Ubuntu的使用和运行全编译的机会很少，所以也是非常宝贵的体验。Chromium OS本身的开发现在也非常流行。低规格的PC也可以运行，基础上网本等支持Windows XP的PC都可以。其实与本系列同时连载的还有Chromium OS活用的文章，但现在与那个时候[①]的编译方法还是有很大的不同。有兴趣的朋友可以去看《Software Design合订本2001-2012》。

编　译

编译环境会把OS完整编译，这是我们想要做到的规格。笔者实际在使用的配置是AMD A8-3820四核CPU，8 GB内存。HDD剩余空间40 GB足够。请使用Ubuntu 12.04的AMD64版本。避免使用日文Remix等安装的i386版本[②]。然后sudo权限也是必需的。这样的话大概就不会有问题了。

首先从准备开发工具开始。请执行代码清单1的命令。

编译Chromium OS，原则上使用专用打包

① 从目前的连载回数就可以知道大概是在三年前。
② 若有i386版本编译也可能通过，但好像会变得比较慢。而且如果没有怎么测试可能会碰到很多陷阱，所以这样做很重要。

工具，安装后，再使用sudo完成剩余工作。把代码清单2的内容做成脚本。

这样编译的准备就完成了。然后，需要决定编译架构。Chromium OS的术语称为“board”，共有三类：

x86-generic …………… i386用
amd64-generic ……… AMD64用
arm-generic …………… ARM32位用

其中，x86-generic最为妥当，所以这里使用x86-generic，请执行如下命令：

```
$ export BOARD=x86-generic
```

取得代码，然后执行如下命令：

```
$ repo init -u https://git.chromium.org/
chromiumos/manifest.git
```

最后会询问用户名和邮件地址，请输入。在这之后就是开始下载。

```
$ repo sync
```

因为会将OS完全下载下来，根据网速不同，有时会花费不少时间。可以边阅读本杂志的其他文章边等待下载完成。如果显示出“Your sources have been sync'd successfully.”，那就说明下载完成了。

终于要执行编译了，请执行如下命令：

```
$ cros_sdk -- ./build_pakages ?board=${BOARD}
```

完成后会显示“INFO   : Elapsed time (build_packages): 16m59s”等所花费的时间。此环境花费了17分钟左右编译完毕。

编译完成后，还要做成镜像。请执行如下命令：

```
$ cors_sdk -- ./build_image ?board=${BOARD}
```

镜像做成后也会显示所花费的时间。此次环境下为“INFO   : Elapsed time (build_image): 10m3s”。

将镜像转送至USB闪存的方法，及虚拟机的镜像转换方法如代码清单3所示，这也是此次环境执行后的案例展示。

向USB闪存转送时，可以将闪存内容全部删除，以确保有4GB以上[3]空间，并将其插入PC。然后执行代码清单4的命令。

③ 做成的镜像文件大小为2.4GB，所以2GB以下明显不够。现在是600日元就可以买到对应USB3.0的4GB USB闪存的时代，所以还是准备一个专用的吧。

▼ 代码清单1　准备开发工具

```
$ sudo apt-get install git-core gitk git-gui subversion curl
$ mkdir chromiumos; cd chromiumos/
$ git clone https://chromiumo.googlesource.com/chromium/tools/depot_tools.git
$ export PATH = "$PATH":`pwd`/depot_tools
```

▼ 代码清单2　开发工具的准备

```
#!/bin/sh
cat > ./sudo_editor <<EOF
#!/bin/sh
echo Defaults !tty_tickets > $1
echo Defaults timestamp_timeout=180 >> $1
EOF
chmod +x ./sudo_editor
sudo EDITOR=./sudo_editor visudo -f /etc/sudoers.d/relax_requirements
```

第二次执行的命令，并不是这里写的，而是执行终端上显示的内容。因为from对象的路径不同，所以执行失败。确认无误输入后，按下Enter键就会显示如下注意事项：

```
WARNING: this will erase all data on /dev/sdb:
BUFFALO USB FLASH Disk,
```

显示内容会因USB闪存型号不同而不同，确认无误后输入“y”，然后按下Enter键。当然不需要输入双引号。这里也会稍微花费点儿时间，最后出现类似dd的执行结果消息，就表示输出完成。

此时可以顺便制作虚拟机镜像，执行如代码清单5所示的命令。

这个也要执行终端显示的内容。虽然按照说明似乎可以制作VMware用的镜像，但实际上制作的却是Qemu/KVM用镜像。做成镜像的地方多少有点儿难以理解，但from对象的文件夹里chromium_qemu_image.bin文件已被生成。笔者使用的是VirtualBox，所以使用了此镜像文件转换后的文件，尝试的过程中没有使用鼠标，所以使用Qemu/KVM是很安全的。

执　行

如果没有安装Qemu/KVM的话，请安装一下。

```
$ sudo apt-get install qumu-kvm
```

执行如代码清单6的命令。

与执行image_to_vm.sh后所显示的消息大概一致，路径的指定稍有不同，请注意。启动后会感觉窗口有些小，可以用鼠标调整为任意大小。

虚拟机的情况下，除了比较慢以外不会出现什么大的问题，如果从USB闪存启动，手持

▼ 代码清单3　转送至USB的方法和虚拟机的镜像转换方法

```
Done. Image(s) created in /mnt/host/source/src/build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1
Developer image created as chromiumos_image.bin
INFO    : Elapsed time(build_image): 10m3s
To copy the image to a USB key, use:
  ./image_to_usb.sh --from=../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1
To convert it to a VMWare image, use
  ./image_to_vm.sh --from = ../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1 ?board=x86-generic
```

▼ 代码清单4　转送至USB闪存

```
$ cd src/scripts/
$ ./image_to_usb.sh ?from = ../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1
```

▼ 代码清单5　虚拟机镜像文件做成

```
$ ./image_to_vm.sh --from=../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1 ?board=x86-generic
```

▼ 代码清单6　执行

```
$ sudo kvm -m 1024 -vga cirrus -pidfile /tmp/kvm.pid -net nic, model=virt | io -net
users,hostfwd=tcp::9222-:22 -had ../build/images/x86-generic/R29-4145.0.2013_ | 05_18_1238-a1/
```

PC不能启动的情况比较多。笔者实验以后的结果如表1所示。

设 定

启动时的设定仅有使用语言和Google账户以及账户所显示的头像。启动后的界面没有壁纸，你可能会觉得不知所措，若点击右下角的钟表会出现“设定”。滚动到最下面，点击显示详细设定。首先将时间的时区设置为东京。然后点击输入语言设置。选中“Mozc（for Japanese Keyboard）”。想变更ATOK键组合的话，可以点击“设定”，将“选择键设置”设置为“ATOK”。Mozc的启动方法为设定画面中显示的Alt+Shift键。

壁纸和Chromium的主题也可以更改，但系统没有预存壁纸，所以需要自己准备[4]。

与设定稍微有些不同，按下Ctrl+Alt+F2可以显示虚拟控制台。返回为Ctrl+Alt+F1。虚拟控制台中会显示用户名等信息。

第二次以后的编译

第二次以后的编译会更加简单，这里就不赘述了，大致执行代码清单7就可以了。

④ Google+上传的照片好像也可以使用。

之后请执行image_to_usb.sh和image_to_vm.sh。

注 意

因为Chromium OS的开发很活跃，所以根据编译时间不同，很有可能会发生失败或者不能启动等问题。编译的方法也可能随之发生变化。虽然如此，但是也没有比这更加简单的方法了。这种情况下，请确认英文版Quick Start Guide[5]。如果还是不能编译成功，说不定过几天再试一次就会编译成功。

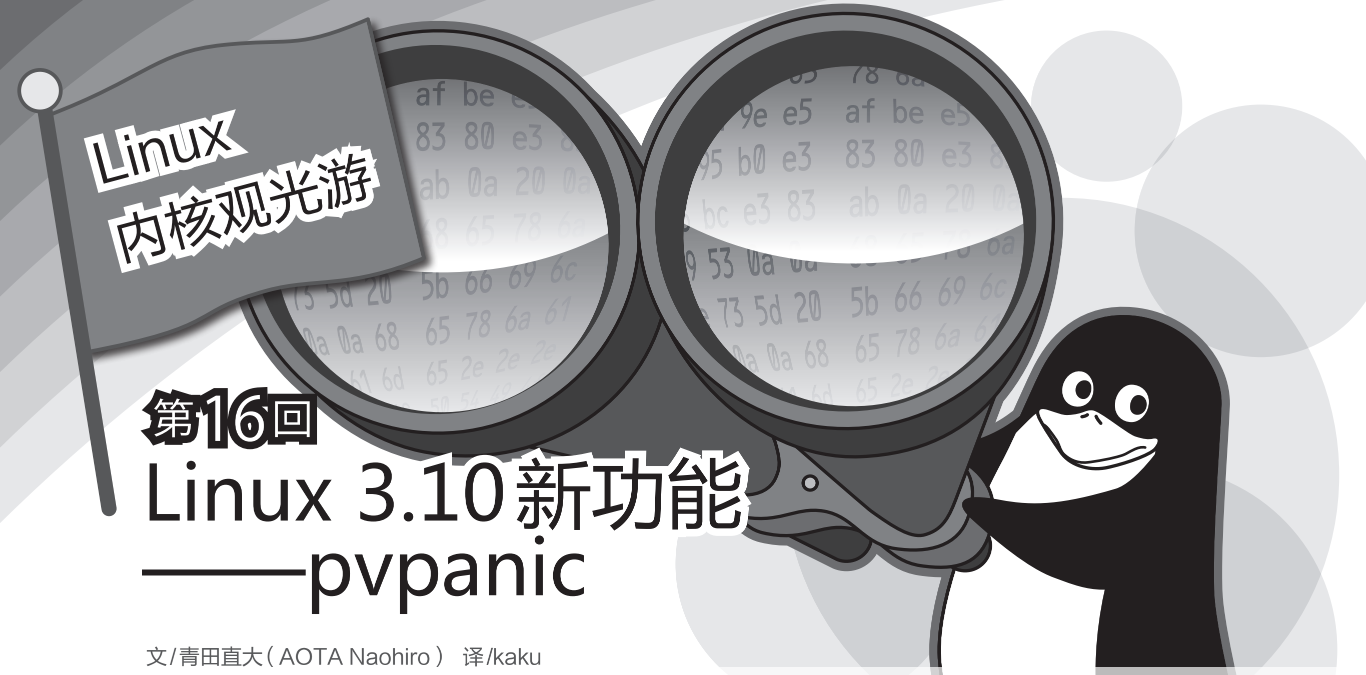
⑤ http://dev.chromium.org/chromium-os/quick-start-guide

▼ 代码清单7　第二次以后的编译

```
$ cd chromiumos/depot_tools
$ git pull
$ cd ../
$ export PATH="$PATH":`pwd`/depot_tools
$ repo sync
$ export BOARD=x86-generic
$ cros_sdk -- ./build_packages ?board=${BOARD}
$ cros_sdk -- ./build_image ?board=${BOARD}
$ cd src/scripts
```

表1　启动状况

发售年代	厂　商	型　号	启动状况	备　注
2008	ASUS	Eee PC 901-16G	×	画面不能显示
2009	ACER	Aspire Timeline AS1410	○	-
2012	ASUS	U24A-PX3210	×	启动了但是触摸屏不能使用
2012	ASUS	X202E-CT987G	×	未能启动



第16回 Linux 3.10新功能——pvpanic

文/青田直大（AOTA Naohiro） 译/kaku

内核3.9发布以后，内核3.10的开发也随之开始。3.0版下的小版本号，不知不觉已到达两位数了，由此也可看出Linux内核开发是相当活跃的。这个月我想来介绍一下Linux 3.10中pvpanic和Android的Synchronization framework等新功能。

pvpanic

pvpanic功能可以在虚拟机客户端的内核发生错误时，向主机端发出通知。若尝试此功能必须有以下环境：

- linux内核3.10以上
- qemu 1.5.0以上
- seabios的git版

Linux内核设定部分如下：

```
Device Drivers --->
    [*] X86 Platform Specific Device Drivers --->
      <M>    pvpanic device support
```

将以上三处安装后（将内核模块安装到客户端的文件系统中），启动客户端的Linux。首先向从QEmu而来的ISA错误通知接口，以及从seabios而来的ACPI的错误通知接口中添加QEMU0001的ID。ACPI的情况下如图1所示，可以确认接口已被添加。接下来向内核读入模块pvpanic后，准备工作就结束了。使用sysrq-trigger来故意使内核发生错误（图2）。

内核错误发生后切换至QEmu的监视器。若使用图形窗口则按Ctrl+Alt+2，使用参数-nographic启动qemu的情况则按Ctrl+a+c切换至QEmu监视器。在监视器中查看虚拟机的状态，就可以在客户端检验出错误信息（图3）。

与此同时，通知事件会将QMP（QEMU Monitor Protocol）协议发生的错误发送出去。qemu加-qmptcp:127.0.0.1:4444,server参数后启动，在qmp-shell① 中指定此地址链接。这是为了和QEmu监视器同样地查看虚拟机

① 从这里可以拿到：http://git.qemu.org/?p=qemugit;a=tree;f=QMP;hb=HEAD

▼ 图1　ACPI的确认和模块读取

```
# strings /sys/firmware/acpi/tables/SSDT|grep QEMU
QEMU0001
# modprobe pvpanic
# lsmod
Module                  Size  Used by
pvpanic                 2017  0
```

▼ 图2　使内核发生错误

```
# echo c > /proc/sysrq-trigger
[  256.428775] SysRq : Trigger a crash
[  256.429307] BUG: unable to handle kernel NULL pointer dereference at           (null)
[  256.429460] IP: [<ffffffff814581b6>] sysrq_handle_crash+0x16/0x20
[  256.429460] PGD 1c122067 PUD 1cbd5067 PMD 0
[  256.429460] Oops: 0002 [#1] SMP
[  256.429460] Modules linked in: pvpanic
[  256.429460] CPU: 0 PID: 776 Comm: bash Not tainted 3.10.0-rc2+ #250
[  256.429460] Hardware name: Bochs Bochs, BIOS Bochs 01/01/2011
[  256.429460] task: ffff88001c090000 ti: ffff88001c054000 task.ti: ffff88001c054000
[  256.429460] RIP: 0010:[<ffffffff814581b6>]  [<ffffffff814581b6>] sysrq_handle_crash+0x16/0x20
[  256.429460] RSP: 0018:ffff88001c055e68  EFLAGS: 00010092
[  256.429460] RAX: 000000000000000f RBX: 0000000000000063 RCX: 00000000000005da
[  256.429460] RDX: 0000000000001099 RSI: 0000000000000046 RDI: 0000000000000063
[  256.429460] RBP: ffff88001c055e68 R08: 0000000000000003 R09: ffffffff81ea8048
[  256.429460] R10: 00000000000001ab R11: 00000000000001aa R12: ffffffff81c93c40
[  256.429460] R13: 0000000000000286 R14: 0000000000000001 R15: 0000000000000000
[  256.429460] FS:  00007f9df51ab700(0000) GS:ffff88001fc00000(0000) knlGS:0000000000000000
[  256.429460] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[  256.429460] CR2: 0000000000000000 CR3: 000000001cf1f000 CR4: 00000000000006f0
[  256.429460] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[  256.429460] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
[  256.429460] Stack:
[  256.429460]  ffff88001c055ea8 ffffffff814589a1 ffff88001c055ee8 0000000000000002
[  256.429460]  fffffffffffffffb 00007f9df51b2000 ffff88001c055f50 0000000000000001
[  256.429460]  ffff88001c055ec8 ffffffff81458a43 ffff88001c055ee8 ffff88001ddb4300
[  256.429460] Call Trace:
[  256.429460]  [<ffffffff814589a1>] __handle_sysrq+0x121/0x190
[  256.429460]  [<ffffffff81458a43>] write_sysrq_trigger+0x33/0x40
[  256.429460]  [<ffffffff812102c3>] proc_reg_write+0x43/0x70
[  256.429460]  [<ffffffff811b207e>] vfs_write+0xce/0x1e0
[  256.429460]  [<ffffffff813b105a>] ? trace_hardirqs_off_thunk+0x3a/0x6c
[  256.429460]  [<ffffffff811b2562>] SyS_write+0x52/0xa0
[  256.429460]  [<ffffffff81721006>] system_call_fastpath+0x1a/0x1f
[  256.429460] Code: 45 01 f4 45 39 65 34 75 e5 4c 89 ef e8 24 f8 ff ff eb db 66 90 0f 1f 44 00 00
55 c7 05 14 fa a4 00 01 00 00 00 48 89 e5 0f ae f8 <c6> 04 25 00 00 00 00 01 5d c3 0f 1f 44 00 00 55
48 89 e5 53 48
[  256.429460] RIP  [<ffffffff814581b6>] sysrq_handle_crash+0x16/0x20
[  256.429460]  RSP <ffff88001c055e68>
[  256.429460] CR2: 0000000000000000
[  256.429460] ---[ end trace 1afcd15abc8fc3e7 ]---
[  256.429460] Kernel panic - not syncing: Fatal exception
```

▼ 图3　ACPI的确认和模块读取

```
QEMU 1.5.50 monitor - type 'help' for more information
(qemu) info status
VM status: paused (guest-panicked)
```

状态，添加运转中的磁盘，关闭虚拟机电源。读取pvnanic模块，可以查看内核发生的错误和通过qmp-shell发送的事件。

图4表示正在发送错误事件GUEST_PANICKED和虚拟机关闭事件STOP。这里关闭虚拟机也是通过pvpanic完成的。

pvpanic的结构

我们先从初始化的部分来看pvpanic的结构。

在QEmu中加入has_pvpanic参数，参数为true的情况下，pvpanic_init()被调用。has_pvpanic在1.5版本的虚拟机中默认为true，除此以外则为false。pvpanic_init()通过isa_create_simple()来注册pvpanic的ISA总线[②]。

虚拟机启动时，pvpanic的ISA总线的初始化函数pvpanic_isa_initfn()被调用[③]。这个函数会准备ISA的I/O端口，而且这个I/O端口的位置会被记录在machine/fw_cfg对象中，并保存至文件名为etc/pvpanic-port的文件中[④]。

Seabios在构建记载ACPI的SSDT（Secondary System Description Table）设备信息表时，会检测是否存在文件etc/pvpanic-port[⑤]。如果这个文件存在，就会将文件中写着的pvpanic的I/O端口号写入，若不存在则将0写入SSDT表的QEMU0001 ID中。这就是虚拟机使用ACPI寻找pvpanic的I/O端口，以及可以从ACPI的SSDT的QEMU0001中取得的原理。

客户端pvpanic的内核模块被加载时，首先查看API的SSDT，以及从QEMU0001取得pvpanic的I/O端口[⑥]。然后，内核错误时做成回调函数pvpanic_panic_notify。

内核错误发生后通过pvpanic_send_event()使用outb()，向I/O端口发送1[⑦]。从这里传送给QEmu的handle_event()（代码清单1）。通过hand_event()从QEmu向QMP发送GUEST_PANICKED事件，虚拟机则变为RUN_STATE_GUEST_PANICKED状态而停止（图5）。

于是QEmu、Seabios、linux内核就这样被巧妙地组合起来通知内核错误。

② http://git.qemu.org/?p=qemu.git;a=commitdiff;h=3ab135f3462af4c523a4b5969f9d6c672ac427a

③ http://git.qemu.org/?qemu.git;a=commitdiff;h=eec3d2adc9 8dd9ef7352823ce6597f88a51cf7cb

④ http://git.qemu.org/?qemu.git;a=commitdiff;h=10a584b2875a391d1036adac18955a892e56f5e3

⑤ http://code.coreboot.org/pseabios/source/tree/master/src/acpi.c

⑥ http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/drivers/platform/x86/pvpanic.c#n93

⑦ http;//git.kernel.org/cgit/linux/kernel/git/torvalds/linux.tit/tree/drivers/platform/x86/pvpanic.c#n59

▼ 图4　qmp-shell的错误通知

```
# qemu -qmp tcp:127.0.0.1:4444,server ...
# qmp-shell  127.0.0.1:4444
Welcome to the QMP low-level shell!
Connected to QEMU 1.5.50

(QEMU)
{u'timestamp': {u'seconds': 1369325618, u'microseconds': 719474}, u'data': {u'action': u'pause'},
u'event': u'GUEST_PANICKED'}
{u'timestamp': {u'seconds': 1369325618, u'microseconds': 953920}, u'event': u'STOP'}
(QEMU) quit
{u'return': {}}
```

Android sync driver

之前加入Android的内核功能合并到了Linux本身中，同样3.10中出自新Android功能的Synchronization framework，以及使用它的Software synchronization objects的实现也合并在其中。

Synchronization framework，从它的名字就可以想象出，其作用就是同步各驱动。但是，仅仅是同步的话，内核中应该已经完全具备了这个功能。Synchronization framework，到底有怎样的特征呢?

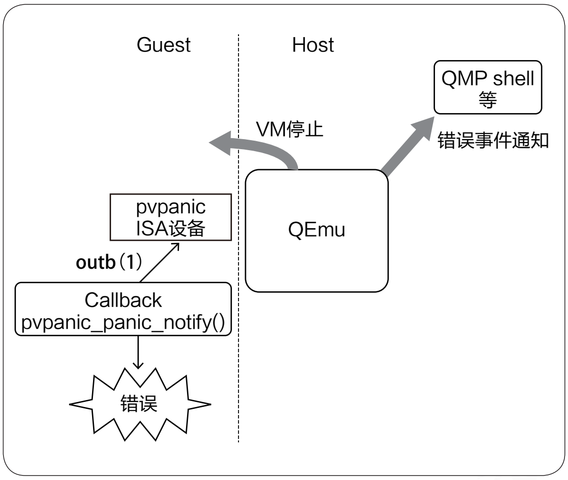
Synchronization framework有这样三个基本的构造：

- timeline
- sync point
- fence

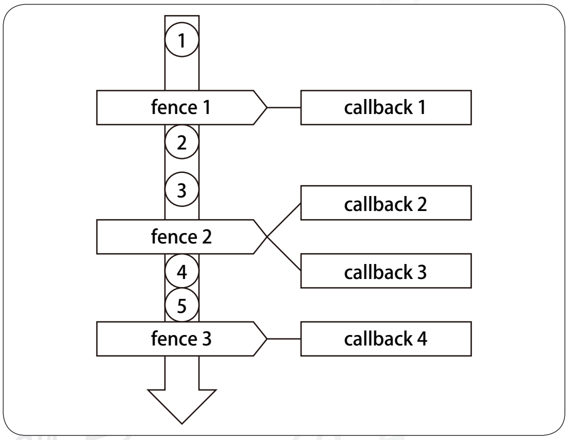
下面为到timeline时的流程。在timeline中设置有多个sync point。与各个sync point对应都有fence，反过来fence对应一个以上的sync point。简单的例子如图6所示。

sync point虽然已发送了signal，但未发送的仍有两个状态。而且，fence追加至对应的文件描述符，对应此文件可以poll或wait（设置暂停），或者设置回调函数。这些

▼ 图5　pvpanic的动作



▼ 图6　timeline的流程



▼ 代码清单1　QEmu的handle_event

```
static void handle_event(int event)
{
    static bool logged;

    if (event & ~PVPANIC_PANICKED && !logged) {
        qemu_log_mask(LOG_GUEST_ERROR, "pvpanic: unknown event %#x. n", event);
        logged = true;
    }

    if (event & PVPANIC_PANICKED) {
        panicked_mon_event("pause");
        vm_stop(RUN_STATE_GUEST_PANICKED);
        return;
    }
}
```

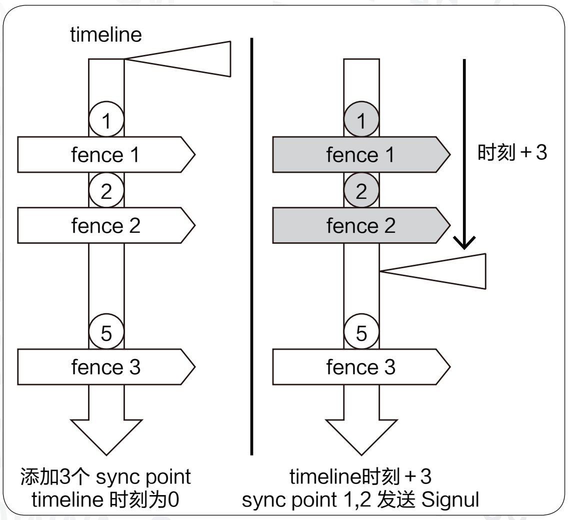
poll、wait、callback的动作，在fence的所有sync point发送signal之后poll、wait结束，callback被执行。拿图6来说，sync point从1到4发送signal的话，callback1到3会被执行但callback4不会被执行。

在这里“信号被发送”的含义是问题所在。实际上这里使用Synchronization framework实现依赖，在做成timeline时可以指定sync point是否发送signal的判断函数。例如，Software synchronization objects用时间这样的对象来检验是否signal被发送。

/dev/sw_sync设备文件open()[⑧]的话，Software synchronization的timeline被做成“时刻”设置为0。这个文件有两个种类的ioctl()可以执行。其中之一为做成指定某个“时刻”的sync point和这个sync point所拥有的fence。另一个仅指定“时刻”的数字就可以了。timeline的“时刻”若在sync point的“时刻”之后，sync point会被发送signal。也就是说，首先在“时刻”1、2、5做成sync point的话会变为如图7左侧所示。然后，试着前进到“时刻”3的话如图7右侧所示，timeline的“时刻”会移动，结果“时刻”1、2的sync point会发送signal。

⑧ 必须将内核设置为SW_SYNC_USERR。

▼ 图7　根据/dev/sw_sync的两个ioctl()



mempressure的导入

从这里开始，介绍3.10加入的几个比较重要的功能。首先是在前面的文章中介绍过的mempressure notification[⑨]，再简单介绍一下。加入这个功能后在memory cgroup中新的memory.pressure_level文件会被添加。为获取通知的eventfd，打开这个memory.pressure_level，把这些描述符和想要获取的pressure level一并写入cgroup.event_control，从eventfd获取对应level的通知。

level有low、medium和critical三种。low会在回收未使用的内存作为预留时发出通知。medium会在回收内存，且所见60%以上不能回收（因为在使用中）时发出通知。这是系统在开始swap，并将正在使用中文件的缓存大致开始清除的时候。因有剩余，所以还有时间调查系统的状态，并检查是否释放应用程序端使用中缓存。critical的所见90%以上内存不能回收时发出通知。若到达这种情况的话swap的I/O系统会变得几乎

⑨ http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=70ddf637eebe47e61fb2be08a59315581b6d2f38

不能运行，要马上使用OOM Killer强制停止进程。如果出现这种情况要马上采取一些措施。

脚本执行功能可自由选择

接下来，如同我们熟知的UNIX脚本，在文件的开头“#!”指定脚本解释器变为嵌入模块而不是内核，也就是说从内核中完全分离了出来[10]。于是，完全不能使用脚本的系统（或者若加入模块中，mount根文件系统之前不能使用的系统）也变得可以使用了，只是在有些地方会不太好用。

randconfig的改善

编辑Linux内核设定项目的程序kconfig也稍有改变。Linux内核设定参数的数量变得很多。可能功能A设置为ON、功能B设置为OFF时编译失败，启动也可能失败。但是，也不可能实现这个设定参数全部组合测试。反之，自由地将设定项目设为ON/OFF的话，可以找出人工操作之外bug的可能性组合。从“make randconfig”自由设定项目为ON/OFF已经可以实现。在3.10中，randconfig的功能中这两个地方都做了改善。

第一，KCONFIG_PROBABILITY环境变量可以随机变更为ON/OFF[11]。目前为止，单纯ON（y）/OFF（n）的话可能性各占50%，组合（y）/模块（m）/无（n）的话可能性为33%/33%/34%，全部选择项可以被均等地选择。比如把CONFIG_PROBABILITY设置为N和0～100的数字，可能变为ON是N%，变为OFF是（100-N）%。其他N:M/N:M:L的设定可能如表1中所列（y:n为ON/OFF的单纯设定项目，y:m:n为组合/模块/无的设定项目）。

第二，环境变量KCONFIG_BANDSEED可以设定随机的种子值[12]。若给予同样的种子值，就可以一直得到同样随机的设定。

	y : n	y : m : n
N	N:100-N	N/2:N/2:100-N
N:M	N+M:100-(N+M)	N:M:100-(N+M)
N:M:L	N:100-N	M:L:100-(M+L)

总结

关于Linux 3.10中加入的功能，本次以pvpanic为主，简要介绍了mempressure和脚本执行功能的改善。有了pvpanic，可以在虚拟机发生错误时发出通知（而且，可以停止虚拟机），利用git bisect可以将内核测试自动化，也可以监视虚拟机运行服务的状态。

⑩ http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=2535d0d723d4d7723b030f39fb350e436bdb983f

⑪ http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=e43956e607692f9b1c710311e4a6591ffba1edf0

⑫ http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=0d8024c6ebadb68f1154377c2e1996b4e649e4c8

jus快讯

Japan UNIX Society

日本UNIX用户协会 http://www.jus.or.jp/
法林 浩之(HOURIN Hiroyuki hourin@suplex.gr.jp)
译/王凤波

jus走过的30年历程

日本UNIX用户协会jus成立于1983年6月，于2013年5月迎来了30周年庆典。因此，这一次我们一改以往向大家报道各种活动的惯例，为大家介绍一下jus走过的30年历程。

成立经过及初期活动

UNIX是在20世纪70年代后半期传入日本的，当时还没有互联网，所以要进行UNIX信息的交流时，用户与用户聚集在一起面对面交谈是最行之有效的交流方式。进入80年代，伴随着UNIX机器的出现，各用户公司成立了独立于销售商之外的UNIX用户团体即jus。成立初期，其具有代表性的活动包括发布有关UNIX研究成果的UNIX专题讨论会以及UNIX机器和软件展销会UNIX Fair等。

对UNIX和Internet发展的贡献

从20世纪80年代后半期到90年代初期，UNIX也开始被积极地应用于商务领域，而jus作为信息交流的平台，其存在的重要性变得更加不可小觑。特别是UNIX Fair展销会，对于当时正着手于UNIX计算机生产和OS开发的各家公司而言，UNIX Fair展销会是各家产品齐聚一堂、相互交流的重要平台。在最繁盛时期，jus会员数达到约2500人，UNIX Fair展销会的入场人数曾达到4万余人。

另外，这一时期正处于日本互联网时代的黎明期。最初是通过UUCP在UNIX计算机之间建立连接从而形成了JUNET（Japan University NETwork），之后JUNET进一步发展而形成了Internet。当时，UNIX Fair是检验各个公司UNIX计算机之间相互连接性的平台。

在这之后，jus继续涉足互联网技术，参加了1996年开始的国际互联网大会Internet Conference（IC）以及次年开始的互联网周Internet Week（IW），2008年起又参加了IPv4地址枯竭工作组。

■ 与Windows共存

20世纪90年代后半期以后，比UNIX计算机更廉价的、面向Windows操作系统的计算机开始普及。UNIX如何与不同架构的Windows实现共存，无论从技术层面还是从商务层面，都成为了一个重要的课题。这一时期的jus曾与Windows用户团体共同举办过专题谈论会，并开展了UNIX Fair的后继活动Network Users’等。

■ 个人学习并使用UNIX的时代

继Windows普及之后，在UNIX领域也出现了运行在个人计算机中的Linux和FreeBSD，迎来了个人拥有UNIX计算机的时代。为了顺应这样的历史潮流，jus于1993年成立了个人计算机环境研究会，并通过开展专题讨论会等活动，为PC-UNIX提供了信息交流的平台。

从1994年起，jus开始每月定期举办UNIX学习会。在那个年代，各种研究讨论会的入场费普遍都是1万多日元，而jus为了让个人能够在工作之余自费参加UNIX学习，破例将入场费定在了1000日元。20年之后的今天，几乎每天都在进行

着由各种社团主办的学习会，而jus则是这些学习会引以为豪的时代先驱。

涉足UNIX以外的领域

■ 对Open Source的贡献

20世纪90年代末，开放源码的概念开始兴起。迄今为止，jus一直通过各种活动与BSD和Linux等团体保持着联系。作为开源企业、社团以及用户的交流平台，jus分别于1999年和2001年举办了开源大会，并于2002年起开始与日本关西地区的社团共同举办了关西开源论坛。

■ 对轻量语言社团的贡献

21世纪初以后，jus与编程语言社团之间的交流变得更加频繁。之前，jus一直通过YARPC等活动与Perl和Ruby等社团保持着交流。在此基础之上，PHP和Python也加入，并于2003年起开始了轻量语言（Lightweight Language）活动。除此之外的其他语言团体也被卷入到了这一历史潮流之中，轻量语言活动作为盛夏的一项重要盛典一直延续至今。

■ 对地方社团的贡献

jus成立之初便开始定期举办关西UNIX研究会，为关西地区UNIX社团的形成做出了贡献。20世纪90年代后，jus又开始在东海地区举办研究会。这两项活动都一直延续到2005年左右。2007年起，继这两项活动之后，jus将举办研究会的范围扩展到日本全国，进一步加强了与各地方社团的交流。

◆ ◆ ◆

jus近几年的活动基本上是固定不变的，全年举办的活动有LL、KOF、IC和IW，定期举办的活动有学习会和研究会。除此之外，还有与gadget社团共同举办的Gadget1活动。从这些活动我们可以看出，近年来，由jus单独运营的活动仅限于学习会之类，与其他社团的共同合作已经开始变得不可或缺。

总 结

通过上述回顾我们可以知道，jus当初是面向UNIX而成立的团体，但在历史的长河中因为其能够弹性地顺应时代的需求，灵活应对各种不断涌现的新概念，所以才能顺利地发展到今天。这是jus没有固步自封或固守某一技术领域，积极主动进取、全力以赴的结果。这也是jus能够持续走过30年历程的秘诀所在。当然，在当时的历史条件下，能够应对各种技术领域的团体只有jus，也是其中一个因素吧。

最近，各种社团已经有能力独立举办各种活动了，所以jus对于社团本身进行的支援活动有所减少。但是我相信，UNIX相关产品今后依然会继续受到人们的欢迎，所以jus也一定会继续在这里扮演好自己的角色，并做出自己应有的贡献。想到这里，让我不禁有些心潮澎湃，希望大家今后继续多多关注jus。

创造互联网服务未来的人们 24

探索CyberAgent公司的网络基础设施（前篇）

采访/撰文：㈱Insight Image　川添贵生（KAWAZOE Takao　mail@insightimage.jp）　　摄影：中村俊哉 NAKAMURA Toshiya
译/王凤波

今年春天，CyberAgent公司构建了自己的私有云环境，从而为拓展新的服务项目奠定了坚实基础。我们就CyberAgent公司的现有网络环境、当前所面临的课题以及今后的发展前景，采访了篠原雅和、高桥哲平和村越俊克这三位工程师。

构建完全支持IPv6的网络基础设施

互联网IP地址资源管理机构IANA（Internet Assigned Number Authority）和IPNIC（Asia Pacific Network Information Centre）所掌管的IPv4地址池资源于2011年相继枯竭。最近，某些电信运营商更以IPv4地址资源不足为由，将公有IP地址的分配列为可选收费服务项目。可见，IPv4地址资源的枯竭已经成为一个非常现实的问题。IPv6曾被视为治本之策而备受期待，但是现实情况却不容乐观，还很难说IPv6已经得到了普及。

但是，对于服务提供商而言，无论IPv6何时正式投入使用，都要未雨绸缪，提前做好准备工作。实际上，CyberAgent公司在东京市内的数据中心为新服务而构建的基础设施中，就实现了对IPv6的全面支持。

“这一次，新的数据中心网络是在完全支持IPv6理念下构建的。对于还需使用IPv4的部分应用，我们采用了IPv4和IPv6共存的双协议栈，除此之外的其他应用基本上都只采用了IPv6地址。实际上，从1年前左右，我们就在原有数据中心构建了IPv6网络环境进行测试，积累了一定的经验，所以这一次IPv6网络的构建还是很顺利的。”（篠原）

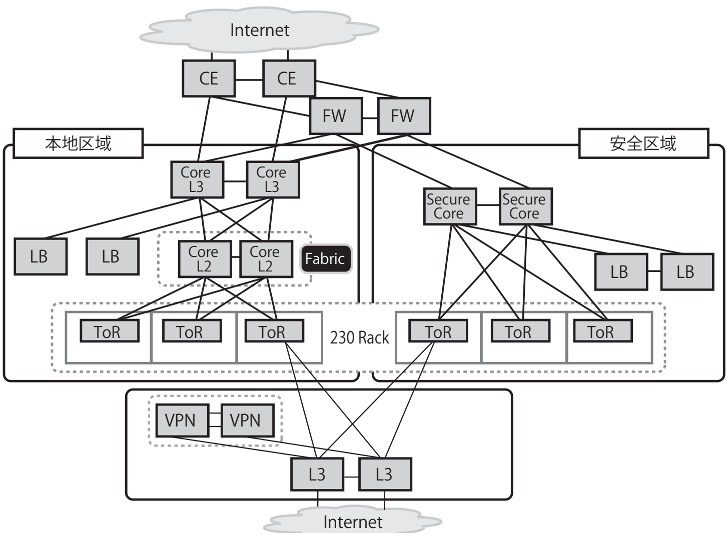


图1　CyberAgent公司新构建的网络结构可以分为提供一般性服务的“本地区域”和处理高机密性数据的“安全区域”这两个部分

在探讨基础设施对于IPv6支持的问题时，最让人头疼的是建成兼顾分配IPv4地址与IPv6地址的双协议栈，还是只分配IPv6地址。对此，CyberAgent公司制定的方针是，除了那些



照片1　村越(左)、哲平(中)、篠原(右)

仅能通过IPv4进行通信的软件所处的节点之外，基本上只分配IPv6地址。究其原因，篠原先生作了如下阐述。

“原因有好几方面，其中之一就是考虑到如果分配了IPv4地址，就需要对其进行管理，从而可能无法充分发挥IPv6的优势。如果对IPv4地址也进行管理，就不可避免地会产生相应的管理成本。这次新网络的构建，因为部分服务器不得已而分配了IPv4地址，所以还不能说我们已经完全削减了IPv4的管理成本。但是，我们把基本上只分配和应用IPv6地址作为了新网络构建的前提条件。”(篠原)

IPv6网络构建中出人意料的盲点

但是，据说在IPv6网络的构建过程中也遇到过令人意想不到的屏障。

“为了实现路由器的冗余，我们采用了VRRP协议(Virtual Router Redundancy Protocol)，但这与RA(Router Advertisement)的兼容性发生了冲突。具体情况是，当发生故障从主机切换到备机再切回到主机时，主机下的节点从主机路由和备机路由都收到了RA，因此，虽然已经切回到了主机，数据包还是被发送到了备机路由上。最终，我们是通过调整VRRP的时钟解决了这一问题，但是在这之前，甚至到完成测试的阶段，我们都一直没有觉察到竟然会发生这样的问题。”(篠原)

关于其他的盲点，村越先生还列举了一个非常出人意料的不支持IPv6的例子。

“由于网络设备对于IPv4的限制，有时系统日志保存地址或者设置信息的导出地址只能设置IPv4地址。网络设备本身支持IPv6，在流量转发功能上并没有问题，但是其管理功能却不支持IPv6。这真是一个让人意想不到的盲点啊。”

另外，据说外部连接并没有采用IPv6直接连接服务器的形式。对此，哲平先生解释道：“我们现在已经做好了随时都可以发送AAAA记录的准备，但是考虑到客户端的处理，现在尚且处于静观其变的状态，要看今后的发展情形而定。”

此次CyberAgent公司新构建的基础设施已经实现了对IPv6的全面支持，但是关于原有基础设施对IPv6的支持，又是如何考虑的呢?哲平先生作出了这样的回答：“原有基础设施，现在是不支持IPv6的，我想也没有这个必要。如果将来有一天，原有基础设施需要支持IPv6的话，我认为直接构建一套新的基础设施就好了。如果强制性地将原有基础设施改建成支持IPv6的话，可能会发生一些意想不到的问题，从这一点来考虑，我想如果确实有这方面需要，还是应该重新构建一套新的基础设施比较好。”

关于本次IPv6网络的构建，篠原先生“在关键部分没有发生任何重大问题”。由此可见，虽然在ISP处理等方面还存在着一些不明了的地方，但是我们已经可以认为，IPv6技术现在已经是一种能够为人们所广泛应用的技术。

下次我们将继续采访CyberAgent公司，就CyberAgent公司对未来网络环境的构想作进一步探讨。



第38回 如何发布放心安全的Android应用

Google Android作为第一个移动设备的开源平台，吸引了很多工程师的关注。学习那些积累了丰富经验的Android工程师分享的技巧和信息，朝着Android的世界大步前进吧！

文/谷口岳　TA-SOFT公司
译/唐洪军

智能手机的普及和安全状况

智能手机近几年得到了十分迅速的普及。和传统手机不同，智能手机用户自己就可以安装软件，十分方便。不过安全方面怎么样呢？用户随身带着手机去到了什么地方，和谁通话了，通讯录里都有什么朋友等，这些私人信息都存储在手机里，并且，应用程序（简称应用）都可以访问这些信息。

恶意应用，当然也可以访问这些私人信息，有些恶意应用目的就是为了盗取联系人信息，但是通讯录、SNS这些应用都需要取得联系人信息，所以从用户的联系人里读取信息的应用并不能都称作恶意软件，有很多应用我们都不能分辨它是为了正常使用读取联系人信息，还是以盗取为目的读取这些信息。这让用户很担心。

这种情况之下，Android应用的开发者能够做些什么，才能给用户提供“放心安全的应用程序”呢？本文将对“安全的应用程序制作方法”和“让用户放心的应用程序发布方法”进行解说。

安全应用程序的制作方法

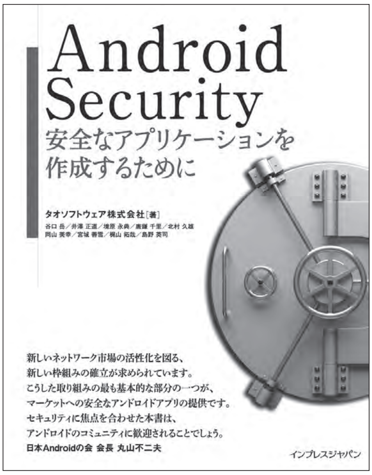
应用里有漏洞的话，用户信息就会被其他恶意应用盗取或者破坏。可惜的是，Android官方文档里面并没有约定的安全文档来供程序员参考。日语资料当中，也只有下面两本书。

- 《Android Security：如何制作安全的应用程序》①（图1）
- 《Android应用的安全设计：安全编码指南》②（图2）

虽然资料只有这两本，但是反过来说，只要读懂这两本书，就可以大体掌握Android的安全事项。希望大家能把这两本书读完，不过在这里只能先作为参考资料介绍给大家。本文主要给读者介绍以下三个Android应用比较容易发生漏洞的地方。

① TA-SOFT公司著作 Impress JAPAN出版。
② 日本智能手机协会（JSSEC）的公开资料。

▼图1 《Android Security：如何制作安全的应用程序》



▼图2 《Android应用的安全设计：安全编码指南》



数据泄露，Android应用内部生成的文件，要设置成只有自己可以读写（API的默认操作）。

需要注意的是，利用外部存储设备（SD卡等）时，因为外部存储设备的特性，即使设置了文件的读写权限，也还是可以被其他应用读写。所以重要的文件不要放在外部存储设备上。

组件的漏洞

Android应用是由多个组件（Activity, Service, BroadcastReciver, Content Provider）一起构成的。组件之间可以互相交换数据，但很多时候，其他应用的组件也可以访问组件的内部数据。

比如说，只在应用内部使用的Content Provider如果被外部应用访问，其他应用就可以读取到内部数据。还有拨打电话时，这个Activity也可能会被外部的应用访问，本来没有权限的应用就能打电话了。为了避免这样的事情，如果这个组件不需要被外部应用访问的话，应该把AndroidManifest.xml的export设定成false。

文件的漏洞

重要文件如果被其他应用读取，就会发生

WebView的漏洞

应用为了显示WEB页面，经常用到WebView。只要给WebView指定好URL，就可以像浏览器一样打开页面，这是一个非常简单、便利的组件。但如果WebView载入了含有恶意JavaScript代码的HTML，一旦JavaScript被执行，数据就会被盗取。为了避免类似事件发生，需要注意以下几点：

1. 使用WebView时，如无必要，需要把JavaScript设置成不可执行。（默认操作）
2. 使用JavaScript时，WebView里载入的HTML不要放在可以被其他应用更改的地方（如外部存储设备等）。
3. 使用JavaScript时，只能载入指定的URL地址，并且用HTTPS协议来通信。

这里讲的几个漏洞，都是比较常见且很容易被发现的。因为设计问题或者一不小心，就

会发生，必须要引起注意。

让用户放心的应用程序发布方法

智能手机里存储着很多类似用户行踪、通话记录的信息，各种应用访问这些信息后，再把这些信息发送到别的地方。然而利用目的不明，没有任何说明就随意取得这些信息的应用越来越多，这让用户觉得很担心。鉴于这种情况，2012年8月，日本的总务省颁布了“智能手机隐私权保护准则”[③]。

智能手机隐私权保护准则的基本要求

这个准则的基本要求是：每个应用都要制定隐私权政策，在读取一些特定信息时，还需要另外取得同意。大家应该看一下这个准则，但实际上它适用于很多智能手机的开发，而且不光针对应用的开发者，对移动通信商、OS的开发公司、应用介绍页面、信息插件的提供者，统统都适用。所以我们公司把只针对Android应用开发人员的部分“Android智能手机隐私权保护准则 by TAO-SOFTWARE[④]”发布在公司的网站上，大家阅读一下这个准则就可以了（图3）。

▼ 图3 《Android智能手机隐私保护准则》



智能手机隐私权保护准则当中，要求那些取得用户信息、进行外部通信和存储的应用，要做到以下两点：

1. 制定应用程序的个人信息保护方案。
2. 取得重要信息时，以对话框的形式告知用户。

在说明这两点之前，先解释一下什么是用户信息。用户信息包括：通讯录等第三方信息、用户的通信服务记录、和用户状态相关的信息等一切与智能手机用户相关的所有信息。它和个人信息保护法里的个人信息是不同的。虽然用户信息和个人信息保护法有关联，但是因为每个人对于“个人信息”的解释不一样，使用“个人信息”这概念，会比较繁琐，所以定义了新的概念“用户信息”。

应用程序的隐私权政策

做应用就要有应用的隐私权政策，现在几乎所有公司的主页上都有隐私权政策，这个和手机应用的隐私权政策是有区别的，开发的应用如果都使用相同的隐私权政策的话，隐私

③ http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000087.html
④ http://www.taosoftware.co.jp/android/android_privacy_policy/

权政策的内容与应用的实际功能很可能有出入，这就有点不准确了，所以每一个应用都必须有自己的隐私权政策。另外，如果应用里有一些应用协议的话，不能把应用协议和隐私权政策写在一起，否则会让人难以理解，最好分成两个文件。

隐私权政策里，需要包含以下8方面的内容：

1. 应用的作者姓名或者是公司名

包括开发该应用的公司名称和联络地址等。

2. 需要取得信息的项目名称

指的是Mail地址，电话号码这样具体的项目，而不是联系人信息那样广泛的定义。

3. 取得方法

是应用自动取得，还是需要用户输入。

4. 明确使用的目的

以应用本身的使用为目的，还是应用以外（广告）的使用为目的，还是要把取得的数据提供给第三方。

5. 通知、公布和取得用户同意的方法，让用户参与的方法

什么时候取得用户同意（应用启动的时候，还是信息取得的时候），取得用户同意的方式是以弹出对话框的形式，还是没有对话框。

用户参与的方法，如果用户想停止信息被利用和取得时，应该如何在应用上操作。应用应该在服务器端删除相关的用户信息，重要的是用户可以随时停止应用取得自己的信息。

6. 有没有外部发送，第三方提供的信息插件

“第三方”是指应用作者以外的，或者是提供服务的公司以外的人，比如说，应用里包含的广告公司、信息收集的公司。如需向第三方公司提供信息（大部分都是广告公司），需要记录以下的几点内容：①使用的收集信息的插件名称，②信息收集插件的作者名称，③取得信息的项目，④使用目的，⑤是否需要向第三方提供信息。但很多时候，有的信息连应用开发人员都不提供。现在由广告公司协会来把这些信息汇总到方案里。再过一段时间，日本会颁布关于广告插件的隐私权政策，可以学习一下这个方案。有一点需要注意的是，使用外国广告模块时，有些信息是很难汇总到方案中的，请参看稍后要讲的第三方模块使用时的注意点。

7. 咨询窗口

咨询窗口的联系地址。

8. 隐私权政策更改时的流程

如需更改隐私权政策，经常有很多公司，会在主页上以“本公司网站通知”的方法来告知用户，在应用中，即使Web页面有所更改，用户也很难注意到，所以不能使用这种方法。隐私权政策中，重要部分变改时，必须在应用内用对话框的形式来通知用户。

另外，在总务省的文件里虽然没有提及，Android系统中，权限许可信息是判断应用特点的重要信息，权限许可和利用的目的记录在方案中比较好，大家可以通过下面的

URL看一下本公司做成的隐私权政策的具体实例。

隐私权政策的实例

- tPacketCapture
  http://www.taosoftware.co.jp/android/packetcapture/
- tSpyChecker
  http://www.taosoftware.co.jp/android/spychecker/

隐私权政策的提示

制定好的隐私权政策放在什么地方，是一件非常重要的事情，它需要满足以下三个条件。

- 应用安装之前，能够查看隐私权政策的概要。
- 应用安装之前，能够查看隐私权政策。
- 从应用里能够查看隐私权政策。

隐私权政策一般都很长，与使用规则差不多，一般人可能都不会认真阅读，因此，在Google Play里发布应用时，以概要的形式记录在应用的说明栏里，不失为一个好方法（图4）。

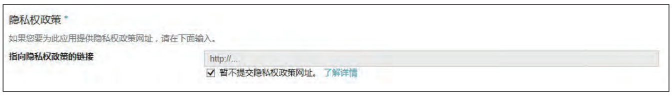
Google Play里，都会有应用的隐私权政策的URL（图5）。它记录在应用的说明页里。在应用的详细画面，可以找到个人信息保护方案的链接（图6）。用户只要一按，就可以显示全部个人信息保护方案（图7）。

另外，应用在使用过程中，有时也需要看一看隐私权政策，这时建议在应用内事先存储好，用菜单和“关于”对话框的形式显示给用户。

▼ 图4　隐私权政策概要的例子

> **隐私权政策概要**
>
> 本应用程序需要对通讯录进行备份，因此需要取得通讯录的全部信息，并将其发送至本公司的服务器。
>
> 这些数据在本应用程序以外，不会以任何其他目的被使用，应用程序里不含有广告，也不会向广告公司发送数据。
>
> 隐私权政策的详细信息，请通过http://www.taosoftware.co.jp/android/packetcapture/#privacy_policy进行查阅，通过此URL可以跳转到开发者信息的隐私权政策页。

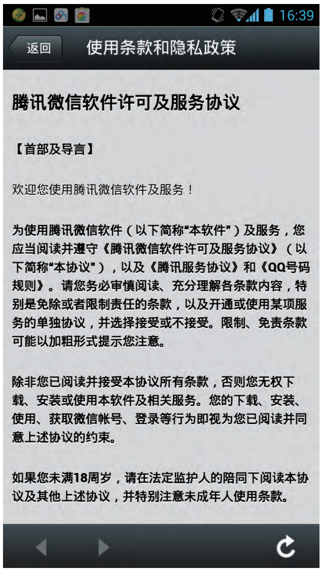
▼ 图5　Google Play Developer Console上的登录画面



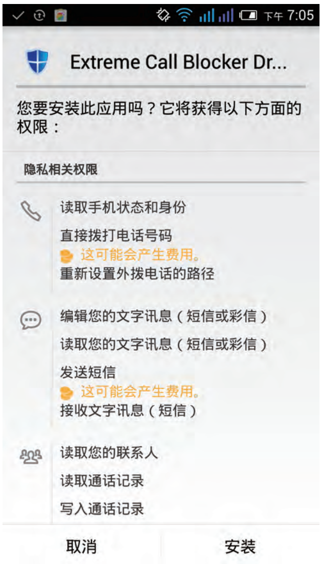
▼ 图6　应用详细页面的隐私权政策链接



▼ 图7　点击之后的提示



▼ 图8　权限许可的确认页面



▼ 图9　弹出对话框示例



同意取得的对话框

取得重要数据的时候，必须要征得用户同意，Andriod应用安装的时候，应该显示权限请求的确认页面。（见图8）

一定要注意的一点是，并不是按了“同意并下载”按钮，就等于同意可以随意取得用户数据了。因为，具体的利用目的，向外部通讯的有无，是否会把信息提供给第三方并没有明确写出来，到底同意了什么，说得并不清楚。需要获得用户同意时，应该像图9那样，在应用内部，用弹出对话框的形式详细地说明之后，才能获得用户的同意。

到底是否需要以弹出对话框的形式来取得用户的同意，一般基于以下两点来判断：

1. 用户信息的性质和种类

2. 用户信息的利用目的

第1条说的是，利用个人信息以及个人识别度较高的用户信息时，需要单独取得同意。简单地说，就是个人信息和能够确定个人信息的信息，如姓名、年龄、通讯录等，或者用户自己很难更改的，比如说IMEI或者Android ID。

第2条说的是，用户很清楚通讯录应用取得联系人的信息是为了应用自身的功能，但是，用户很难理解，应用以广告、掌握用户使用情况这样的目的取得自己的信息。所以，应用除自身功能需要以外，想取得用户信息时，就需要得到用户的单独同意。

需要强调的是，利用IMEI、Android ID和应用本身功能以外，即便是很微小的信息，都需要得到用户的单独同意。

把权限的请求范围尽可能地缩小

隐私权政策和取得用户同意的对话框是很重要的，但在Android系统中，因权限不同，应用能做的事情是不一样的。有取得联系人信息权限的应用，还是没有取得权限的应用，哪一个更让人放心呢？当然是没有权限的那个应用。

Google Play的应用之中，获得本身不必要权限的应用有很多。其原因是开发时，参考了需要取得权限的既存功能，或参考的信息本身就有错且没有及时发现，或者根本没有好好理解，使用的是类库的API等。

所以，请务必把权限的请求范围控制在自己需要的最小范围内。

使用第三方模块时的注意点

最后，要注意一下应用里面别人做的东西，以前发生过因为Facebook SDK有漏洞，导致所有使用了Facebook SDK的应用都具有同样漏洞的事件。如果有源代码的话，一定要看一看源代码。另外，虽然几乎所有的广告插件都没有代码，但有一些广告插件具有恶意应用的功能，一旦植入了这样的广告插件，这款应用也会被认为是恶意应用。一般来说，给PC买软件的时候，如果感染了病毒，软件开发公司必须得承担责任。同样，如果发布了含有病毒的应用，应用的开发公司肯定是有责任的，因为它给很多客户带来了麻烦。

很多广告应用都是要取得用户信息的。得到越多的用户信息，广告收入带来的收益就越大。虽然广告收益是比较重要的收入来源，但那种乱七八糟的广告模块还是不要使用为好。

Column

Android应用漏洞诊断
Web服务

Tao RiskFinder

如同刚才讲的使用第三方插件，现在安卓应用开发所用的类库，很多也包含外部插件。这种情况下，很可能无意间就使用了，如许可证书有问题，就可能使用了危险的插件。另外，Android应用的漏洞是比较大的问题，定制Andriod应用的公司，很难发现来历不明的插件或漏洞。基于此，本公司开发了漏洞检查工具Tao RiskFinder，现在还仅限于公司团体使用，但是如果对此有兴趣的个人，也可以尝试使用，这款应用将来要做成个人开发者也能使用的形式，希望这个举措能让现在的应用变得更完善。

Tao RiskFinder
http://www.taosoftware.co.jp/services/riskfinder/index.html

总　结

发布放心安全的应用，是很费工夫的。有时会听到这样的说法："只是想单纯地开发一个应用"，"担心一旦告诉用户应用需要取得哪些用户信息，用户一害怕，就不用你的应用了"。但笔者认为，做了东西让别人用，不管能不能得到报酬，开发者都是有责任的。不要做那种会给用户带来麻烦的、有漏洞的软件。另外，写一些对自己不利的东西，可能的确让人有所抵触，但如果真的觉得自己做的东西很重要的话，就不要有所隐瞒，堂堂正正地发布吧！

谷口岳（Tao Software公司CEO）

Android OS发布之后，就开始从事Android的应用开发，并利用BLOG和演讲的形式，进行Android的推广宣传活动。现在转职到专业开发Android的公司，从事委托开发和顾问的工作。

温故知新
IT的古老传说

第24回

6809/OS-9 /6829 MMU



竹冈尚三　TAKEOKA Shouzou　take@takeoka.net　译/王凤波



引子

6809（MC6809）是摩托罗拉公司（如今的飞思卡尔）于1979年继6800（MC6800）之后推出的、拥有丰富寻址模式的8位CPU[①]，它具备与C语言中*P++和*--P相类似的指令。

由于6809有着非常强大的寻址模式，使得用汇编程序编写位置无关代码（Position Independent Code）变得非常容易。程序代码附近的常量按照PC（程序计数器）相对寻址模式访问，函数中的本地变量存储在栈中，按照堆栈指标相对寻址模式访问。由操作系统创建的（用于全局变量的）数据存储空间的开始位置，将在创建进程之时存放于U寄存器（用户栈指标）中，所以按照U寄存器相对寻址模式访问（栈空间也是创建进程之时由操作系统创建的）。通过编写这样的程序，可自动实现独立进程间的可重入（reentrant），多个进程之间可以共享内存中的同一段二进制代码。

6809虽然是8位机，但是如上所述，因其具备强大的寄存器相对寻址功能，所以使得使用汇编程序编写代码变得毫不费力。



6809与OS–9

大约在1980年以后，出现了与6809对应的操作系统，即OS-9（之后被称为OS-9/6809）。该OS-9操作系统是在参考UNIX的基础上，使用6809汇编程序编写的多用户多任务操作系统，包括Level1和Level2两个版本。Level1运行在不需要特殊硬件支持的64位内存空间中。Level2运行在配备有6809外置MMU（具体请参照下文）的硬件之上。OS-9 Level2是以映射器（Mapper）为前提的，而OS-9 Level1则是在没有这种地址转换结构的硬件下实现多任务处理的，所以在可执行二进制代码的处理上下了很大的功夫。



OS–9 Level1

因为OS-9 Level1是运行在8位CPU即6809的64KB空间中（包括I/O在内），所以其模块化实现得比较彻底，设备驱动以及重要的系统功能也都实现了模块化。大部分系统模块都被称为“管理器”，并被命名为“xxxman”。虽然内存空间狭小，但是因为OS-9 Level1不读取那些不必要的模块，所以即使是结构非常简单的模块也能够确保正常运行。

OS-9是参考UNIX编写而成的，所以其访问控制也全部是通过文件访问权限来实现的。它的任务与UNIX一样被称为“进程”，进程之间的通信是通过与UNIX类似的管道PIPE来实现的。PIPE通过pipeman来

① 当时被称为“终极8位CPU”。

实现，并通过上层IOMan与文件一起实现统一化管理。OS-9的命令行也实现了与UNIX shell一样的语法，从而确保了可以使用“>”、“<”、“¦”等语法进行IO重定向，并使进程输入输出连接的建立成为可能。

OS-9要求可执行二进制代码中的位置无关代码（PIC）程序。位置无关代码无论在内存中的任何位置执行都能够确保程序的正常运行。另外，OS-9的进程代码还同时要求实现可重入性（Reentrancy）。

应用于OS-9的BASIC语言和C语言编译器因为推出的时间较早，所以没有什么特别问题。关于那些不支持OS-9的编译器以及汇编源代码，因U寄存器作为临时存储（scratch pad）使用得非常频繁，并且允许全局变量的访问直接载入变量地址，所以事实上这些编译器以及汇编源代码未能得到广泛应用。

内存中二进制的复用

OS-9的节约精神主要还是体现在创建进程时查找可执行二进制代码上。在OS-9/6089中，并没有对于执行结束后的程序代码进行管理，而是将其作为堆空间的备用空间而置之不理。OS-9会在某一进程开始时，首先以字节为单位扫描所有内存空间，查找二进制代码的首部，然后根据该首部对二进制代码的所有字节进行CRC（循环冗余校验码）计算。如果该CRC值正确，也就是说若该二进制代码在内存中未遭到损坏，且正好是所需要的二进制代码，则重复使用。如果内存中未查找到所需要的二进制代码，则从文件系统将二进制代码载入到内存中。

上次为大家介绍的MP/M是由载入器在载入内存时执行PRL（Page Relocatable）格式的地址解析，而OS-9则采用位置无关代码PIC，在执行之前便完成了对于地址的计算。

6829 MMU

曾经有一款6809配套的摩托罗拉标准产品，即MMU(存储管理器，Memory Management Unit)，这是一种大规模集成电路（LSI），被称为6829。6829只进行内存映射（以页面为单位进行地址转换），不能检测出未载入的页面。6829以多任务页面切换为前提，不同页面使用不同内存映射，从而避免任务之间的内存冲突。

为了正确执行上述处理，6829的控制寄存器不能被用户程序所操控。另外，控制由操作系统转移至用户任务时，也同时需要将操作系统的内存映射切换至用户任务的内存映射。为了正确实现这一点，6829配备了一个叫做fuse register的寄存器，在控制即将由操作系统转交给用户任务之前，操作系统会在fuse register寄存器中设置一个计时时间，当经过这一计时时间时，6829的内存映射便发生转换，从而实现无法再继续访问控制寄存器。

接下来则需要等到发生中断之时（包括软中断）才能转换为操作系统内存映射，并对控制寄存器进行操作。这是纯粹的配备有6829 MMU的6809系统，而OS-9 Level2则多运行于那些未配备6829而使用了专有MMU（映射器）的系统之上。

◆

关于OS-9/6809，在其成熟期曾运行于富士通FM-11、FM-7/8、日立MB-S1等在日本非常流行的计算机上，在美国也曾运行于TRS80 Color Computer（CoCo）等比较廉价的计算机上，因此了解OS-9/6809的人应该不在少数。

关于OS-9/68000，也推出过用于没有配备MMU的68000之上的版本，曾运行于夏普X68000之上，另外还曾长期应用于CD-i等系统之上。

看完了

如果您对本书内容有疑问，可发邮件至contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱：ebook@turingbook.com。

在这里可以找到我们：

微博 @图灵教育 : 好书、活动每日播报
微博 @图灵社区 : 电子书和好文章的消息
微博 @图灵新知 : 图灵教育的科普小组
微信 图灵访谈 : ituring_interview，讲述码农精彩人生
微信 图灵教育 : turingbooks